

CDC® CYBER 200 OPERATING SYSTEM VERSION 1

FOR USE WITH
CYBER 200 SERIES
COMPUTER SYSTEM

Volume 1 of 2

REFERENCE MANUAL



CONTROL STATEMENT SUMMARY

The following control statements call CYBER 200 utilities. The notation used in the control statement formats is described in the section Notations Used in this Manual. The page number corresponding to each format refers to the page containing the control statement description.

	<u>Page No.</u>
ATTACH, $\left\{ \begin{array}{c} \text{lfm-list} \\ * \end{array} \right\}$, WAIT=waitopt.	4-3
AUDIT, PN=pkid-list, PF=lfm-list, UN=userno, POOL=pl-list, OP=opts, DT=mmddy, TM=hhmm, LO=x, OU=lfm/len/dc.	4-3
COMMENT. message	4-6
COMPARE, alfn, blfn, L=len, A=aadr, B=badr, N=lt.	4-6
COPY, inlfm, outlfm, L=len, I=inadr, O=outadr, PACK=packid.	4-7
DEFINE, lfm/len, ACCESS=acs, TYPE=typ, SECURITY=lvl, PACK=packid, NOEXTEND, NOSEGMENT, RT=rt, MNR=mnr, MXR=mxr, PC=pc, RMK=rmk.	4-7
DEBUG, fname, I=iname, O=oname/olen.	6-1
DUMP, dropfile.	6-10
DUMPF, DD=device, VSN=id-list, DE=density, RE=days, UN=userno, POOL=pl-list, PN=pkid-list, PF=lfm-list, OP=opts, DT=mmddy, TM=hhmm, LO=x, OU=lfm/len/dc.	4-9
EDITPUB, $\left\{ \begin{array}{c} L \\ D=lfm-list \end{array} \right\}$, N=lfm-list, P=lfm-list, VRI=index.	4-12
EXIT.	4-12
FILES, lfm-list, PUBLIC= $\left\{ \begin{array}{c} * \\ lfm-list \end{array} \right\}$, PRIVATE= $\left\{ \begin{array}{c} * \\ lfm-list \end{array} \right\}$, POOL=poolname, lfm-list, L=lfm.	4-13
GIVE, $\left\{ \begin{array}{c} * \\ lfm-list \end{array} \right\}$, $\left\{ \begin{array}{c} U=newown \\ POOL=poolname, SHARE=perm \end{array} \right\}$.	4-14
LOAD, lfm-list, CNTROLEE=lfm/len, CDF=dlen, OUTPUT=lfm/len, LIBRARY=lib-list, EQUATE=sub.name, ENTRY=ept, DEBUG=mod-list, VR=string, ORIGIN=bitadr, GRSP=list,bitadr, GRLP=list,bitadr, GROS=com-list,bitadr, GROL=com-list,bitadr, GRLPALL=Δ, DSA=bitadr, LO=x.	4-15
LOADPF, DD=device, VSN=id-list, DE=density, UN=userno, POOL=pl-list, PF=lfm-list, OP=opts, DT=mmddy, TM=hhmm, PN=pkid-list, LO=x, OU=lfm/len/dc.	4-18
LOOK, fname, I=iname, L=oname/olen/disp.	6-6
NORERUN.	4-21

CDC® CYBER 200 OPERATING SYSTEM VERSION 1

**FOR USE WITH
CYBER 200 SERIES
COMPUTER SYSTEM**

Volume 1 of 2

REFERENCE MANUAL



[illegible]

Address comments concerning this manual to:

**Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112**

or use Comment Sheet in the back of this manual.

All rights reserved

Printed in the United States of America

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Front Cover	-	4-25	C	8-27	C	9-38	C	Comment Sheet	C
Inside Front Cover	C	4-26	C	8-28	C	9-39	C	Inside Back Cover	C
Title Page	-	4-27	C	8-29	C	9-40	C	Back Cover	-
ii	C	4-28	C	8-30	C	9-41	C		
iii/iv	C	4-29	C	8-31	C	9-42	C		
v/vi	C	4-30	C	8-32	C	9-43	C		
vii	C	4-31	C	8-33	C	9-44	C		
viii	C	5-1	C	8-34	C	9-45	C		
ix	C	5-2	C	8-35	C	9-46	C		
x	C	5-3	B	8-36	C	9-47	C		
xi	A	5-4	C	8-37	C	9-48	C		
1-1	C	5-5	C	8-38	C	9-49	C		
1-2	C	5-6	C	8-39	C	9-50	C		
1-3	C	5-7	C	8-40	C	9-51	C		
1-4	B	5-8	C	8-41	C	9-52	C		
2-1	B	5-9	C	8-42	C	9-53	C		
2-2	B	5-10	B	8-43	C	9-54	C		
2-3	C	5-11	C	8-44	C	A-1	B		
2-4	B	5-12	C	8-45	C	A-2	B		
2-5	B	5-13	C	8-46	C	A-3	B		
2-6	B	5-14	B	8-47	C	B-1	C		
2-7	C	6-1	C	8-48	C	B-2	C		
2-8	B	6-2	C	9-1	C	B-3	C		
3-1	C	6-3	C	9-2	B	B-4	C		
3-2	C	6-4	B	9-3	C	B-5	C		
3-3	C	6-5	C	9-4	C	B-6	C		
3-4	C	6-6	C	9-5	C	B-7	C		
3-5	C	6-7	B	9-6	C	B-8	C		
3-6	C	6-8	C	9-7	C	B-9	C		
3-7	C	6-9	C	9-8	C	B-10	C		
3-8	C	6-10	C	9-9	C	B-11	C		
3-9	C	7-1	C	9-10	C	B-12	C		
3-10	C	7-2	C	9-11	C	B-13	C		
3-11	C	8-1	C	9-12	C	B-14	C		
4-1	C	8-2	C	9-13	C	B-15	C		
4-2	C	8-3	C	9-14	C	B-16	C		
4-3	C	8-4	C	9-15	C	B-17	C		
4-4	C	8-5	C	9-16	C	B-18	C		
4-5	C	8-6	C	9-17	C	B-19	C		
4-6	C	8-7	C	9-18	C	B-20	C		
4-7	C	8-8	C	9-19	C	B-21	C		
4-8	C	8-9	C	9-20	C	B-22	C		
4-9	C	8-10	C	9-21	C	B-23	C		
4-10	C	8-11	C	9-22	C	B-24	C		
4-11	C	8-12	C	9-23	C	B-25	C		
4-12	C	8-13	C	9-24	C	B-26	C		
4-13	C	8-14	C	9-25	C	B-27	C		
4-14	C	8-15	C	9-26	C	B-28	C		
4-15	C	8-16	C	9-27	C	B-29	C		
4-16	C	8-17	C	9-28	C	C-1	C		
4-17	C	8-18	C	9-29	C	C-2	C		
4-18	C	8-19	C	9-30	C	C-3	C		
4-19	C	8-20	C	9-31	C	C-4	B		
4-20	C	8-21	C	9-32	C	Index-1	C		
4-21	C	8-22	C	9-33	C	Index-2	C		
4-22	C	8-23	C	9-34	C	Index-3	C		
4-23	C	8-24	C	9-35	C	Index-4	C		
4-24	C	8-25	C	9-36	C	Index-5	C		
		8-26	C	9-37	C	Index-6	C		

PREFACE

This manual describes the CONTROL DATA® CYBER 200 Operating System for the CONTROL DATA CYBER 200 Series Computer System. The CYBER 200 configuration supports either a CONTROL DATA CYBER 200 Link Station running under control of the NOS/BE or SCOPE 3.4 Operating System, or a CONTROL DATA CYBER 200 Access Station running under control of the NOS 1 Operating System.

This manual is published in two volumes:

- Volume 1 is written for applications programmers who will be accessing CYBER 200 Operating System capabilities through batch jobs or interactive terminals. A detailed knowledge of basic programming principles is assumed.

It describes system hardware and software, explains CYBER 200 file concepts and task execution, and provides formats for calling system utilities and subroutines.

- Volume 2 is written for systems programmers who will be modifying or expanding the capabilities of the CYBER 200 Operating System, and who wish to do so by writing programs that communicate directly with other programs and with the virtual system part of the operating system. All virtual system messages available for writing such programs, as well as some of the system tables referred to by these messages, are described. It is assumed that the reader has some understanding of the principles of operating systems in general.

Related information can be found in the following publications.

<u>Control Data Publication</u>	<u>Publication Number</u>
CYBER 200 Operating System Reference Manual Volume 2	60457010

<u>Control Data Publication</u>	<u>Publication Number</u>
CYBER 203 Computer System Hardware Reference Manual	60256010
CYBER 205 Computer System Hardware Reference Manual	60256020
CYBER 200 Operating System Operator's Guide	60457030
CYBER 200 Operating System Installation Handbook	60457020
CYBER 200 Access Station Reference Manual	60452800
CYBER 200 Link Reference Manual	60457060
CYBER 200 Link Operator's Guide	60457070
CYBER 200 FORTRAN Language Reference Manual	60457040
CYBER 200 Assembler Reference Manual	60457050

Control Data manuals can be ordered from:

Literature Distribution Services
STP005
308 North Dale Street
St. Paul, Minnesota 55102

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

1. INTRODUCTION TO CYBER 200 OS	1-1	Card Formats	3-2
System Configuration	1-1	ASCII Coded Cards	3-2
Central Processor Unit	1-1	80-Column Binary Cards	3-2
Central Memory	1-1	Job Structure	3-3
Peripheral Stations	1-1	Job Processing	3-3
Maintenance Control Unit	1-2	Interactive System Access	3-4
High Capacity Disk (HCD) Station	1-2	Request Lines	3-5
Magnetic Tape Station	1-2	Private File Execution	3-6
Coupling Station	1-2	Control Statement Execution	3-7
Service Station	1-2	Message Interrupts	3-7
Unit Record Station	1-2	Accounting	3-7
Operating System	1-2	Job Scheduling	3-8
Resident System	1-2	Resource Allocation	3-8
Virtual System Tasks	1-2	Error Processing	3-9
Privileged User Tasks	1-2	Abnormal Termination Control	3-9
Peripheral Operating System	1-3	ATC Interrupt Subroutine	3-9
OPERATOR Task	1-3	Enabling and Disabling ATC	3-11
System Usage	1-3		
Virtual Memory Addressing	1-3		
Register File	1-4		
CYBER 200 Comparison	1-4		
		4. CONTROL STATEMENTS	4-1
2. FILE CONCEPTS	2-1	Batch Job Control Statement Format	4-1
Mass Storage Files	2-1	Interactive Utility Execution	4-1
File Duration	2-1	ATTACH - Attach Permanent Files	4-3
Permanent Files	2-1	AUDIT - List File Information	4-3
Local Files	2-1	COMMENT - Send Message to Dayfile	4-6
Scratch Files	2-1	COMPARE - Compare File Contents	4-6
File I/O	2-1	COPY - Copy File	4-7
Drop Files	2-2	DEFINE - Create Permanent File or	
Write-Temporary Files	2-2	Make Local File Permanent	4-7
File Access Controls	2-2	DUMPF - Dump Files	4-9
File Security Levels	2-2	EDITPUB - Add or Destroy Public File	4-12
File Access Modes	2-2	EXIT - Set Abnormal Termination Path	4-12
File Ownership	2-3	FILES - List Files	4-13
Private Files	2-3	GIVE - Change File Owner	4-14
Pool Files	2-3	LOAD - Create Controllee File	4-15
Public Files	2-3	LOADPF - Reload Files	4-18
File Structure	2-5	NORERUN - Set Norerun Status	4-21
File Space Allocation	2-5	OLE - Edit Object Libraries	4-21
Logical Record Formats	2-5	Pool File Utilities	4-21
ANSI Fixed Length (F)		PACCESS Pool Utility	4-23
Record Format	2-6	PATTACH Pool Utility	4-23
Record Mark Delimited (R)		PCREATE Pool Utility	4-23
Record Format	2-6	PDELETE Pool Utility	4-23
Undefined Structure (U)		PDESTROY Pool Utility	4-23
Record Format	2-6	PDETACH Pool Utility	4-23
Control Word Delimited (W)		PFILES Pool Utility	4-24
Record Format	2-6	PURGE - Evict Permanent or Pool Files	4-24
Output Files	2-7	READCC - Read Alternate Control Card File	4-25
Punch Files	2-7	REQUEST - Create Local File	4-25
Print Files	2-7	RERUN - Set Rerun Status	4-27
Print Control Characters	2-8	RESOURCE - Set Job Resource Limits	4-27
Magnetic Tape Files	2-8	RETURN - Evict Local Files or Detach	
		Permanent Files	4-28
3. TASK EXECUTION	3-1	ROUTE - Specify File Disposition	4-28
Batch System Access	3-1	SET - Change Memory Limits	4-28
Unit Record Station Batch Input	3-1	SWITCH - Change File Characteristics	4-31
Separator Cards	3-2	TV - Set Threshold Value	4-31
		5. UPDATE	5-1
		Examples	5-1

General Processing	5-2	Q5DESBIF - Destroy Batch Input File	8-16
Update Mode and Files	5-2	Q5DISAMI - Disable Message Interrupts	8-16
Input File	5-4	Q5DISATI - Disable Abnormal Termination	
New Program Library	5-4	Control	8-16
Source File	5-4	Q5DMPACT - Dump Cumulative Accounting	
Old Program Library	5-4	Buffer	8-18
Compile File	5-4	Q5ENAMI - Enable Message Interrupts	8-18
List File	5-4	Q5ENATI - Enable Abnormal Termination	
Creation of Program Library	5-4	Control	8-19
Card Identification	5-5	Q5GETACT - Get Resource Usage Statistics	8-19
Correction Run	5-6	Q5GETCTS - Get Controllee Termination	
Deck List and Directory Order	5-6	Status	8-19
PURGE and YANK Directives	5-6	Q5GETHP - Get Invisible Package	8-22
Overlapping Corrections	5-7	Q5GETIRF - Get Register File	8-22
Update Directives	5-7	Q5GETLP - Get Large Page Limits	8-23
ADDFILE Directive	5-7	Q5GETMCE - Get Message from Controllee	8-23
CALL Directive	5-8	Q5GETMCR - Get Message from Controller	8-25
COMDECK Directive	5-8	Q5GETMOP - Get Message from Operator	8-25
COMPILE Directive	5-8	Q5GETMPG - Get Minus Page	8-27
DECK Directive	5-9	Q5GETPFI - Get Pack Label and File Index	8-27
DELETE Directive	5-9	Q5GETTL - Get Time Limit	8-28
IDENT Directive	5-9	Q5GETTN - Get Task Attributes	8-28
INSERT Directive	5-10	Q5GETUID - Get User Number	8-29
PURDECK Directive	5-10	Q5INIT - Initialize Controllee	8-29
PURGE Directive	5-10	Q5INITCH - Initialize Controllee Chair	8-30
READ Directive	5-10	Q5LFIPOL - List Pool File Indices	8-31
YANK Directive	5-11	Q5LFIPRI - List Private File Indices	8-33
YANKDECK Directive	5-11	Q5LFIPUB - List Public File Indices	8-34
/ Comment Directive	5-12	Q5LSTBUT - List Bank Update Table	8-34
Update Control Statement	5-12	Q5LSTCH - List Controllee Chain	8-34
		Q5LSTSTB - List Statistics Buffer	8-38
		Q5LSTTCB - List Timecard Buffer	8-38
		Q5MSGCTR - Message Control	8-39
		Q5RECALL - Suspend Task Execution	8-39
6. DEBUGGING	6-1	Q5RFI - Return from Interrupt Subroutine	8-40
DEBUG	6-1	Q5RUNBIF - Rerun Batch Input File	8-41
DEBUG Control Statement	6-1	Q5SETLP - Change Current Large Page Limit	8-41
DEBUG Directives	6-2	Q5SNDMCE - Send Message to Controllee	8-42
Dump or Display Directives	6-3	Q5SNDMCR - Send Message to Controller	8-42
Register Directives	6-4	Q5SNDMDMF - Send Message to Dayfile	8-43
Alter Memory Directives	6-4	Q5SNDMJJC - Send Message to Job Controller	8-43
Program Control Directives	6-5	Q5SNDMOP - Send Message to Operator	8-43
LOOK	6-6	Q5SNDSTR - Start Controllee Execution	8-45
LOOK Control Statement	6-6	Q5TERM - Terminate Task	8-45
LOOK Directives	6-6	Q5TERMCE - Disconnect Controllee	8-45
SEARCH Directive	6-7	Q5TIME - Get System Time	8-45
Disposition of Directive Output	6-8	Q5VRACC - Change Accounting Rate	8-48
Display and Dump Directives	6-8		
Directives for Entering Values	6-9		
Declaration of Directive			
Address Type	6-9		
DUMP	6-10	9. SYSTEM INTERFACE LANGUAGE	
		(I/O CALLS)	9-1
7. CHECKPOINT/RESTART	7-1	Overview	9-1
CHKPNT - Checkpoint Call	7-1	File Information Table (FIT)	9-1
Restart	7-2	Tape Label Processing	9-6
		SIL I/O Calls	9-6
8. SYSTEM INTERFACE LANGUAGE	8-1	Q5ATTACH - Attach Permanent File	9-6
(NON-I/O CALLS)		Q5CHANGE - Change File Attributes	9-9
Overview	8-1	Q5CHECK - Check I/O Request Status	9-10
SIL Error Processing	8-2	Q5CLOSE - Close File	9-10
SIL Call Format	8-2	Q5DEFINE - Define Permanent File	9-12
No Operation Keywords	8-3	Q5ENDPAR - Write Partition Delimiter	9-15
SIL Non-I/O Calls	8-3	Q5GENFIT - Generate FIT	9-15
Q5ADVISE - Advise System of Virtual Space		Q5GETFIL - Open or Create and Open File	9-18
Requirements	8-3	Q5GETFIT - Get FIT Field Values	9-22
Q5CPUTIM - Get CPU Time	8-5	Q5GETN - Read Partition	9-25
Q5DCDDST - Decode Disk Status Table	8-5	Q5GETP - Read Partial Partition	9-25
Q5DCDMSC - Decode Miscellaneous Table	8-5	Q5GIVE - Give File Ownership	9-25
Q5DCDPFI - Decode Pack File Index	8-10	Q5MAPIN - Map In Virtual Space	9-29
Q5DCDPLB - Decode Pack Label	8-15	Q5MAPOUT - Map Out Virtual Space	9-30
		Q5OPEN - Open File	9-31
		Q5PATAch - Attach Pool	9-35
		Q5PCREAT - Create Pool	9-35
		Q5PDESTR - Destroy Pool	9-36

Q5PDTACH - Detach Pool
 Q5PGRACC - Grant Access to Pool
 Q5POOLS - List Pools
 Q5PREACC - Remove Access to Pool
 Q5PURGE - Purge File
 Q5PUSERL - List Users with Access to Pool
 Q5PUTN - Write Partition
 Q5PUTP - Write Partial Partition
 Q5READ - Read Block
 Q5REDUCE - Reduce File Space

9-36
 9-36
 9-36
 9-38
 9-38
 9-40
 9-40
 9-41
 9-42
 9-44

Q5RETFIT - Return FIT
 Q5RETURN - Return File
 Q5REWIND - Rewind File
 Q5ROUTE - Route File
 Q5RQUEST - Request Local or Tape File
 Q5SETFIT - Set FIT Field Values
 Q5SKIP - Skip Partition
 Q5STATUS - Get File Status
 Q5WRITE - Write Block

9-44
 9-45
 9-45
 9-46
 9-47
 9-50
 9-50
 9-53
 9-53

APPENDIXES

A. CHARACTER SET
 B. DIAGNOSTICS

A-1
 B-1

C. GLOSSARY

C-1

INDEX

FIGURES

1-1 CYBER 200 Minimum Configuration
 2-1 File Ownership
 2-2 Control Word Format
 3-1 STORE Card Format
 3-2 Example of Batch Deck
 3-3 80-Column Binary Format
 3-4 Example of Batch Job
 3-5 LOGON Format
 3-6 Interactive Task Call Format
 3-7 Example of Interactive LOAD Call
 3-8 Interrupt Subroutine Header
 4-1 ATTACH Control Statement Format
 4-2 AUDIT Control Statement Format
 4-3 AUDIT Sample Output
 4-4 COMMENT Control Statement Format
 4-5 COMPARE Control Statement Format
 4-6 COPY Control Statement Format
 4-7 DEFINE Control Statement Format
 4-8 Directory/Dumped File Format
 4-9 DUMPF Control Statement Format
 4-10 EDITPUB Control Statement Format
 4-11 EXIT Control Statement Format
 4-12 FILES Control Statement Format
 4-13 FILES Sample Output
 4-14 GIVE Control Statement Format
 4-15 LOAD Control Statement Format
 4-16 Virtual Code File Format
 4-17 LOADPF Control Statement Format
 4-18 NORERUN Control Statement Format
 4-19 NORERUN/RERUN Example
 4-20 OLE Control Statement Format
 4-21 PACCESS Control Statement Format
 4-22 PATTACH Control Statement Format
 4-23 PCREATE Control Statement Format
 4-24 PDELETE Control Statement Format
 4-25 PDESTROY Control Statement Format
 4-26 PDETACH Control Statement Format
 4-27 PFILES Control Statement Format
 4-28 PFILES Sample Output
 4-29 PURGE Control Statement Format
 4-30 READCC Control Statement Format
 4-31 REQUEST Control Statement Format
 4-32 RERUN Control Statement Format
 4-33 RESOURCE Control Statement Format
 4-34 RETURN Control Statement Format
 4-35 ROUTE Control Statement Format
 4-36 SET Control Statement Format
 4-37 SWITCH Control Statement Format
 4-38 TV Control Statement Format

1-1
 2-4
 2-7
 3-1
 3-2
 3-3
 3-3
 3-5
 3-6
 3-7
 3-9
 4-3
 4-3
 4-5
 4-6
 4-6
 4-7
 4-8
 4-9
 4-10
 4-12
 4-13
 4-13
 4-14
 4-15
 4-16
 4-18
 4-19
 4-21
 4-22
 4-22
 4-23
 4-23
 4-23
 4-23
 4-24
 4-24
 4-24
 4-24
 4-24
 4-25
 4-25
 4-27
 4-27
 4-28
 4-29
 4-30
 4-31
 4-31

5-1 Typical Update Creation Run
 5-2 Typical Update Correction Run
 5-3 Card Identifier Expansion
 5-4 ADDFILE Directive Format
 5-5 CALL Directive Format
 5-6 COMDECK Directive Format
 5-7 COMPILE Directive Format
 5-8 DECK Directive Format
 5-9 DELETE Directive Format
 5-10 IDENT Directive Format
 5-11 INSERT Directive Format
 5-12 PURDECK Directive Format
 5-13 PURGE Directive Format
 5-14 READ Directive Format
 5-15 YANK Directive Format
 5-16 YANKDECK Directive Format
 5-17 / Comment Directive Format
 5-18 Update Control Statement Format
 6-1 DEBUG Control Statement Format
 6-2 DEBUG Directives
 6-3 Sample DEBUG Directives
 6-4 LOOK Control Statement Format
 6-5 Sample LOOK Directives
 6-6 DUMP Control Statement Format
 7-1 CHPNT Subroutine Format
 8-1 Q5ADVISE Call Format
 8-2 Q5CPUTIM Call Format
 8-3 Q5DCDDST Call Format
 8-4 Q5DCDMSC Call Format
 8-5 Q5DCDPFI Call Format
 8-6 Q5DCDPLB Call Format
 8-7 Q5DESBIF Call Format
 8-8 Q5DISAMI Call Format
 8-9 Q5DISATI Call Format
 8-10 Q5DMPACT Call Format
 8-11 Q5ENAMI Call Format
 8-12 Q5ENATI Call Format
 8-13 Q5GETACT Subroutine
 8-14 Q5GETCTS Call Format
 8-15 Q5GETIIP Call Format
 8-16 Q5GETIRF Call Format
 8-17 Q5GETLP Call Format
 8-18 Q5GETMCE Call Format
 8-19 Q5GETMCR Call Format
 8-20 Q5GETMOP Call Format
 8-21 Q5GETMPG Call Format
 8-22 Q5GETPFI Call Format
 8-23 Q5GETTL Call Format
 8-24 Q5GETTN Call Format

5-1
 5-2
 5-5
 5-7
 5-8
 5-8
 5-8
 5-9
 5-9
 5-10
 5-10
 5-10
 5-11
 5-11
 5-11
 5-11
 5-12
 5-12
 6-1
 6-2
 6-2
 6-6
 6-6
 6-10
 7-1
 8-4
 8-5
 8-6
 8-7
 8-10
 8-15
 8-16
 8-17
 8-17
 8-18
 8-18
 8-19
 8-20
 8-21
 8-22
 8-22
 8-23
 8-24
 8-25
 8-26
 8-27
 8-27
 8-28
 8-28

8-25	Q5GETUID Call Format	8-29	9-8	Q5GETFIL Call Format	9-18
8-26	Q5INIT Call Format	8-29	9-9	Q5GETFIT Call Format	9-22
8-27	Q5INITCH Call Format	8-30	9-10	Q5GETN Call Format	9-26
8-28	Q5LFIPOL Call Format	8-31	9-11	Q5GETP Call Format	9-27
8-29	Q5LFIPRI Call Format	8-33	9-12	Q5GIVE Call Format	9-27
8-30	Q5LFIPUB Call Format	8-35	9-13	Q5MAPIN Call Format	9-29
8-31	Q5LSTBUT Call Format	8-36	9-14	Q5MAPOUT Call Format	9-30
8-32	Q5LSTCH Call Format	8-37	9-15	Q5OPEN Call Format	9-31
8-33	Q5LSTSTB Call Format	8-38	9-16	Q5PATAch Call Format	9-35
8-34	Q5LSTTCB Call Format	8-38	9-17	Q5PCREAT Call Format	9-35
8-35	Q5MSGCTR Call Format	8-39	9-18	Q5PDESTR Call Format	9-36
8-36	Q5RECALL Call Format	8-39	9-19	Q5PDTACH Call Format	9-36
8-37	Q5RFI Call Format	8-40	9-20	Q5PGRACC Call Format	9-37
8-38	Q5RUNBIF Call Format	8-41	9-21	Pool List Entry Format	9-37
8-39	Q5SETLP Call Format	8-41	9-22	Q5POOLS Call Format	9-37
8-40	Q5SNDMCE Call Format	8-42	9-23	Q5PREACC Call Format	9-38
8-41	Q5SNDMCR Call Format	8-43	9-24	Q5PURGE Call Format	9-39
8-42	Q5SNDMDMF Call Format	8-44	9-25	Q5PUSERL Call Format	9-40
8-43	Q5SNDMJJC Call Format	8-44	9-26	Q5PUTN Call Format	9-41
8-44	Q5SNDMOP Call Format	8-45	9-27	Q5PUTP Call Format	9-42
8-45	Q5SNDSTR Call Format	8-46	9-28	Q5READ Call Format	9-43
8-46	Q5TERM Call Format	8-46	9-29	Q5REDUCE Call Format	9-44
8-47	Q5TERMCE Call Format	8-47	9-30	Q5RETFIT Call Format	9-44
8-48	Q5TIME Call Format	8-47	9-31	Q5RETURN Call Format	9-45
8-49	Q5VRACC Call Format	8-48	9-32	Q5REWIND Call Format	9-45
9-1	Q5ATTACH Call Format	9-8	9-33	Q5ROUTE Call Format	9-46
9-2	Q5CHANGE Call Format	9-9	9-34	Q5RQUEST Call Format	9-48
9-3	Q5CHECK Call Format	9-11	9-35	Q5SETFIT Call Format	9-50
9-4	Q5CLOSE Call Format	9-12	9-36	Q5SKIP Call Format	9-52
9-5	Q5DEFINE Call Format	9-13	9-37	Q5STATUS Call Format	9-53
9-6	Q5ENDPAR Call Format	9-15	9-38	Q5WRITE Call Format	9-54
9-7	Q5GENFIT Call Format	9-16			

TABLES

3-1	Program States	3-6	5-2	Summary of Update Directives	5-3
3-2	System Error Codes	3-10	5-3	File Contents and Update Mode	5-9
4-1	Control Statement Functions	4-2	8-1	Message Redirection	8-40
4-2	Interaction of UN and POOL Parameters for AUDIT, DUMPF, and LOADPF	4-5	9-1	FIT Format	9-2
5-1	Summary of Update Call Parameters	5-2	9-2	ANSI Tape Label Formats	9-7

NOTATIONS USED IN THIS MANUAL

UPPERCASE	Words or character strings that must be entered as shown. They must be spelled correctly including any = or / shown.	{ }	Portion of a format in which only one of the vertically stacked items can be used. The braces are editorial conventions only; they are not part of the format.
<u>UNDERLINED</u> <u>UPPERCASE</u>	Words or character strings that can be abbreviated to the number of underlined characters.	...	Repetition indicator. The portion of the format immediately preceding can be repeated at programmer option.
Lowercase words	Generic terms which represent the parameters or character strings supplied by the programmer. When generic terms are repeated in a format, a number or letter might be appended.	Δ	Blank indicator. In a format, this character indicates that a blank or space should appear.
[] Brackets	Optional portion of a format. All parameters enclosed within the brackets can be omitted at programmer option. The brackets are editorial conventions only; they are not part of the format.	#	Numbers used in this manual are decimal unless noted as hexadecimal. Hexadecimal numbers are prefixed by the # character.

Punctuation characters shown within the formats are required unless the text indicates another punctuation character can be substituted.

The CYBER 200 Operating System (CYBER 200 OS) controls a CYBER 200 Computer System. This section gives a general description of CYBER 200 hardware and an overview of the CYBER 200 OS.

The CYBER 200 system must have a maintenance control unit (MCU) and a high capacity disk (HCD) station. To perform I/O, the system must be connected to a front-end processor through a coupling station or to a unit record station through a service station.

SYSTEM CONFIGURATION

The CYBER 200 computer system consists of a central processing unit (CPU), central memory, and peripheral stations. Figure 1-1 shows a CYBER 200 system configuration.

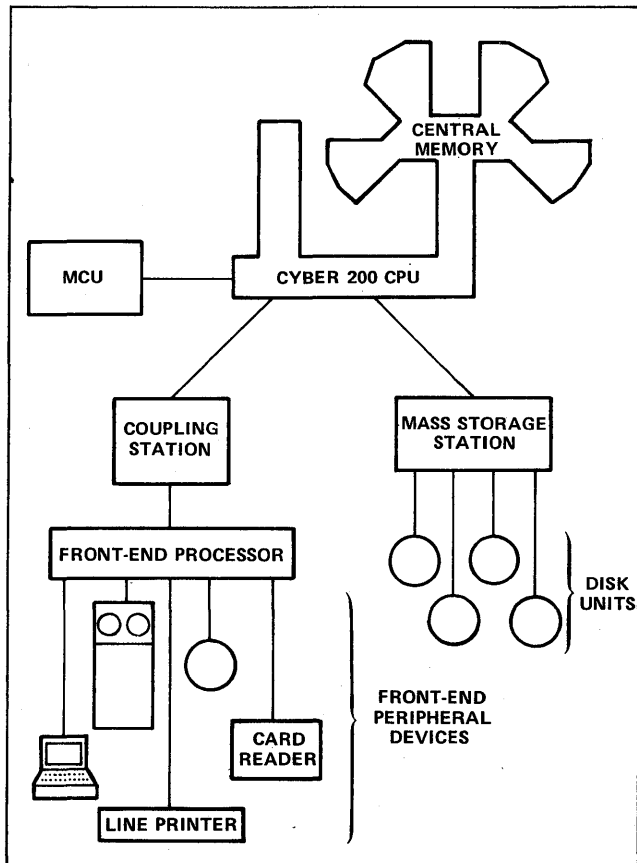


Figure 1-1. CYBER 200 Minimum Configuration

The CYBER 200 Model 203 system can have one-half, one, or two million 64-bit words of central memory; it has 12 input/output (I/O) channels. The CYBER 200 Model 205 can have up to four million words of central memory and up to 16 I/O channels.

CENTRAL PROCESSOR UNIT

The CYBER 200 CPU contains a vector processor, a scalar processor, and the I/O channels. The vector processor performs vector instructions (instructions that use streams of operands to produce streams of results). The scalar processor performs scalar instructions (instructions that produce one result) and directs vector processing and central memory data transfers. An I/O channel controls data communication between the scalar processor and a station.

CENTRAL MEMORY

The CYBER 200 is a virtual memory machine. This means that the user can consider his program space to be virtually unlimited. Program space (virtual space) is associated (mapped in) to central memory and disk storage (physical space). During execution, a program and its data may not be entirely in central memory. When an address in virtual space is referenced whose associated physical space is not currently in central memory, CYBER 200 OS automatically brings the associated physical space in from disk storage.

This process of transferring data only when referenced is called paging or implicit I/O. The blocks of data transferred in and out are called pages. A block of 512 words is sometimes called a small page. A large page is 128 blocks or 65 536 words.

PERIPHERAL STATIONS

The CPU I/O channels communicate with peripheral stations. Each station is a separate processor. The following are the station types.

- Maintenance control unit (MCU).
- High capacity disk (HCD) station.
- Magnetic tape station.
- Coupling station.
- Service station.
- Unit record station.

Maintenance Control Unit

Each CYBER 200 system configuration must have an MCU to deadstart and recover the operating system and to monitor hardware performance. An MCU station consists of a standard station control unit (including a microdrum and display console), card reader, disk storage system, printer, and CPU maintenance hardware interface.

For information on deadstarting and recovering the system from the MCU, the user should refer to the CYBER 200 OS Operator's Guide.

High Capacity Disk (HCD) Station

The HCD station provides on-line mass storage for the CYBER 200 system. Each HCD station consists of a stored program minicomputer (Station Control Unit) with one or two Station Buffer Units (SBUs), each containing 32 K 16-bit words of buffer memory, a microdrum, a CRT console, from one to four channels attached to CPU I/O channels, from one to eight 819 disk drives, and from one to four 7639 controllers.

The 819 disk drive uses a nonremovable disk pack. It has two models, the 819-11 for single-density recording and the 819-21 for double-density recording.

Magnetic Tape Station

The magnetic tape station provides on-line tape access. The station can connect to 657 or 659 tape units.

Coupling Station

The CDC 65209-1 Coupling Station connects the CYBER 200 system to a CDC CYBER 170, CDC CYBER 70, or CDC 6000 Series computer system which acts as a front-end processor for the CYBER 200. The coupling station consists of a station control unit, 8 K of 16-bit memory, a microdrum, a keyboard/display, and a compatible interface to allow direct connection to a data channel. All communication between the CYBER 200 and its front-end is through the coupling station.

Service Station

The service station connects a unit record station and/or a front-end processor to the CYBER 200 system. A service station is not required in the CYBER 200 configuration because a coupling station can connect a front-end processor with unit record equipment to the CYBER 200 system.

Unit Record Station

The unit record station is the interface between a service station and unit record equipment such as 405 card readers, 415 card punches, and/or 512 line printers. A unit record station is not required in the CYBER 200 configuration because unit record equipment is accessible through a front-end processor.

OPERATING SYSTEM

The CYBER 200 operating system has three parts, the resident system, virtual system tasks, and privileged user tasks. The peripheral stations are controlled by the peripheral operating system.

The central operating system requires at least one large page for execution.

RESIDENT SYSTEM

The central operating system and the peripheral operating system each consists of a resident system and a nonresident set of callable tasks. Various portions of the system communicate through messages.

The resident system runs in monitor mode; it is always resident in main memory and references memory by physical addresses, rather than virtual addresses. When the CPU is in monitor mode, interrupts are inhibited, and some additional instructions are enabled.

The resident central operating system has two parts: KERNEL, responsible for time-slicing and message handling, and PAGER, responsible for memory management and page swapping.

All access interrupts, as well as certain messages dealing with memory allocation, are passed to PAGER by KERNEL. PAGER dynamically allocates both large and small pages and performs all required implicit input/output necessary to free memory pages and obtain the pages causing access interrupts.

PAGER determines dynamically which pages of a user's virtual address space have the most activity. These pages define the working set of a program at that time.

VIRTUAL SYSTEM TASKS

The virtual system tasks run in job mode and reference memory by virtual address. They communicate with the resident system via reserved messages and can modify system tables. The virtual system performs resource allocation, file management, explicit input and output, and terminal message processing functions.

PRIVILEGED USER TASKS

Privileged user tasks run under privileged user numbers and can issue privileged and nonprivileged system calls. Unlike virtual system tasks, privileged user tasks cannot modify system tables directly.

Because a privileged user can make most resident system calls, such users are able to perform some tasks for the virtual system. This results in a reduction of virtual system overhead and frees the virtual system to process other functions. Tasks such as handling input and output files and operator communication are currently done by privileged user tasks.

PERIPHERAL OPERATING SYSTEM

The peripheral operating system runs in the station processors. The peripheral system for all station processors has two parts:

- A resident system called NUCLEUS, common to all stations.
- A set of overlays for performing tasks for the individual stations.

NUCLEUS is controlled by SCANNER. SCANNER uses scan bits and an associated table in determining which routines to call. If a particular routine is not in core, a resident overlay driver calls in the routine from the station's microdrum.

NUCLEUS uses a set of nonresident tasks to control peripheral equipment. Operating system tasks for each station processor are stored on its own microdrum.

NUCLEUS consists of diagnostic routines, a system peripheral deadstart program, driver programs for the microdrum and keyboard/display, an organizational program, programs to manage the system overlay mechanism, and SCANNER, which is the main control and organizational program. Nonresident tasks are concentrated into larger processing routines to facilitate on-line error handling and maintenance procedures common to all stations. Further, station functions are grouped into different systems, minimizing the number of system tables. Any one system contains only those routines necessary to its operation.

Operator Task

The CYBER 200 system does not have a special system console. Any terminal logged in with the operator's user number and executing the OPERATOR task can be considered as the system operator's console. Only one terminal can be the system operator console at any one time. The site defines the operator user number.

Once logged in at the system console, the terminal remains in this mode until the operator terminates the OPERATOR task or disconnects the terminal. The operator communicates interactively with the OPERATOR program to request information from the system, respond to user requests for equipment, make general announcements, and so forth.

A detailed description of system operator capabilities is given in the CYBER 200 OS Operator's Guide.

SYSTEM USAGE

Users can access the CYBER 200 system in either batch mode or interactive mode. In either case, the user must enter a legitimate user number and account identifier defined by operations personnel at the CYBER 200 site.

VIRTUAL MEMORY ADDRESSING

Virtual addresses are contained in a 48-bit format. When 512-word pages are addressed, the virtual page identifier is contained in 33 of the 48 bits; for large pages, the virtual page identifier requires only 26 of the 48 bits. Because unused virtual space imposes no burden on the system, the user can organize program addresses in almost any convenient manner.

Virtual addresses are the program addresses that the programmer assumes exist. Virtual addresses are translated into physical memory addresses as needed. The system keeps track of the relationship between physical memory addresses and virtual memory addresses through a translator called a page table. Each entry in the page table contains the virtual page address and the corresponding physical memory address, together with the allowed access mode and other control information. A successful association between a virtual address and an entry in the page table causes that entry to be moved to the head of the table; all entries in between are moved down by one place. In CYBER 200, the first 16 entries in the table are kept in high-speed registers; the registers are examined in parallel with a simultaneous associative compare. An unsuccessful compare results in a sequential search through the remainder of the table held in main memory. Addresses of infrequently used pages automatically float to the end of the table.

If an address has no entry in the page table, the program requesting the address is interrupted. Normally, the system provides the space addressed by requesting the page moved to central memory from the disk. The program continues processing from the point of interruption. The user is unaware of these interruptions.

The virtual memory and the associated paging scheme of the operating system mean that the programmer does not have to break programs into overlays or segments to fit them into central memory. The operating system manages the allocation of storage between central memory and the disk, moves information from the disk to central memory, and translates virtual memory addresses to physical addresses in central memory. However, central memory is where a program executes, and, so, must be taken into consideration when the logical flow of the program is developed.

An efficiently coded application has its data organized so that it can take advantage of the streaming and vector processing capabilities of the CYBER 200. It also has a well-behaved working set which resides in central memory such that the program can efficiently migrate through the working set.

The user should consider whether large pages or small pages are used for the program and data as the programmer determines the block size for a given application.

To be executable, a program must use virtual addressing. Data files can also be defined by a set of virtual addresses. Each active program in the system executes in its own virtual address space; system hardware and software protects the program from other users.

REGISTER FILE

The CPU contains a file of 256 64-bit word addressable registers used for instruction and operand addressing, indexing, field length counts, and as a source or destination for register-type instructions. The register file is accessible to assembly language programs and to FORTRAN language programs which use special calls. Its contents are, by convention, divided into several areas that can be used to pass parameters to another routine, access data for programs, trace execution, and hold constants. The contents of the register file are dumped to an output file as part of abnormal job termination, and a similar dump can be obtained during program execution through the debugging facilities available in the system.

Register file conventions are described in appendix D of volume 2.

CYBER 200 COMPARISON

Significant differences exist between the hardware and software of the CYBER 200 and any of the front-end systems. Some of the differences that affect applications programs are described below.

- CYBER 200 memory words are 64 bits.
- CYBER 200 uses a hexadecimal, rather than an octal, number system. The hexadecimal number sequence is: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- CYBER 200 character data is a subset of 8-bit ASCII, not 6-bit display code. Appendix A contains character code tables.
- Because the CYBER 200 operating system supports virtual memory, programs do not require overlays or segmentation. No overlay or segmentation facilities exist; however, the programmer writing an efficient program is aware of the program flow and develops the application so that the logical flow of the program lends itself to a well-behaved working set.
- CYBER 200 batch job decks are similar, although not identical, to batch jobs processed by the other CYBER computer systems. In particular, all batch jobs running under a single user number are processed sequentially, not in parallel.
- A mass storage file can reside in up to four noncontiguous physical areas of mass storage called segments. Each segment is a contiguous area; all areas are on the same disk.
- CYBER 200 OS is a task-oriented, rather than a job-oriented operating system. Jobs, as such, are known only to the batch processor which initiates execution, in sequence, of each task specified in the job file. Similarly, the interactive processor associates each task it initiates only with the logged-in terminal from which it received the request.

A file is a collection of data accessible by name. File names are 1 to 8 characters long. User-created file names must begin with a letter. System-created file names can begin with any character.

MASS STORAGE FILES

For a virtual memory machine, mass storage (disk) files are essential to system operation. All files in central memory are allocated corresponding space on disk. Data is automatically paged in and out between central memory and disk as required by program execution.

The CYBER 200 OS recognizes files accessed by virtual addressing and by physical addressing (virtual files and physical files). A virtual file is prefaced by a 512-word block containing control information used by the operating system. This preface is known as the minus page. A physical file does not have a minus page.

Only virtual code files that have been processed by the loader are in executable format. A file in executable format is called a controllee file. The loader generates the minus page information for the virtual code file. For more information, refer to the LOAD utility description in section 4.

Both virtual and physical files can be accessed via implicit or explicit I/O.

A detailed explanation of the minus page and other system tables can be found in volume 2.

FILE DURATION

User mass storage files are either scratch, local, or permanent. Scratch files terminate at the end of the program that uses them. Local files terminate at job end. Permanent files are stored until their owner explicitly destroys them.

Permanent Files

The term permanent file implies ownership category private. Permanent files exist until they are explicitly destroyed. Pool and public files must also be explicitly destroyed but are not called permanent files.

A user's permanent files are brought into a system table of active files known as FILEI as soon as the user becomes active in the system by logging on an interactive terminal or running a batch job. Permanent files must then be attached in order to be accessed. To avoid conflicts between jobs, only one suffix can attach a particular permanent file at a time.

Local Files

Files the operating system uses to execute a program are all local files. A program can also request creation of local files.

Local files exist only at the suffix level and are at the same search hierarchy level as attached permanent files. Local files are created and used only for the duration of the terminal session or the batch job. Local files created with the same name but under different suffixes do not conflict, and names of local files do not conflict with the names of unattached permanent files. Unless explicitly made permanent by the user, local files are destroyed at the end of a batch job or an interactive terminal session.

Scratch Files

Scratch files can be created only by a user program. When an executing program terminates normally, all scratch files are destroyed unless they are open to other executing programs of that user. When the operating system terminates the program or the program terminates and saves its drop file, scratch files are saved as local files. Closing a scratch file destroys it. All scratch files have read and write access.

FILE I/O

CYBER 200 allows two modes of I/O from central memory, explicit and implicit.

Explicit I/O uses buffers within the program space. It provides a conventional manner of data transfer between central memory and mass storage or tape. With explicit input/output, more than one page can be transferred between the buffer and a storage device with a single system request. On the other hand, explicit I/O requires more system action than implicit I/O; thus, explicit I/O should be used for large files that transfer many pages at one time.

Implicit I/O does not use program buffers. Information transfers directly to a disk from its current location in central memory. Implicit I/O should be used in tasks that access small files or that access the same part of a file many times.

Implicit I/O occurs when the user references data or code not in central memory. If the virtual page containing the data or code has been previously associated with physical (mass storage) space, the system transfers the data from disk to central memory. If a virtual-to-physical relationship has not been established previously, the system defines the virtual page in free space. The free space is associated with physical space in the drop file.

Drop Files

A drop file is a file that the system creates when a program is put into execution. It contains any modified pages of the program file, any free space attached, and any read-only data space defined to have temporary write access.

When either physical or virtual files are opened for explicit I/O, the system makes entries in a table known as the bound explicit map, which is part of the drop file. These entries are used by the system routines that process explicit I/O requests.

When a file is opened for implicit I/O, the mapping information provided by the user is placed in a table known as the bound implicit map. Map entries in the bound implicit map (part of the drop file) relate a set of virtual addresses to a set of mass storage addresses allocated for the file. The maps have an entry for every discontinuity in virtual address space.

The drop file map is constructed on a page-by-page basis and is of finite size. Attempting to add a page to a full drop file map is a fatal error. To avoid this difficulty, users desiring large blocks of virtual space not represented in some file should create a virtual file and map it into the desired space (refer to section 9).

The system creates the drop file name from the source file name. It shifts the source file name right two characters and enters two digits as the first two characters of the drop file name. The number farthest to the left corresponds to the suffix of the logged-on user. (If the suffix is A, the number is 1; if the suffix is B, number is 2, and so forth.) The second number corresponds to the controllee level number of the program, as follows:

- 1 Batch processor or virtual system interactive processor
- 2 Program initiated by level 1 program
- 3 Program initiated by level 2 program
- 4 Program initiated by level 3 program
- 5 Program initiated by level 4 program

For example, a task named MYTASK01 running under control of the batch processor under suffix D has a drop file named 42MYTASK.

The length of the drop file is taken from the length specified in the file index table for the file. If zero has been specified there, the length of the drop file is taken from the length specified in the minus page of the source file. If zero has been specified there as well, then the length of the drop file is determined from the source file size and drop file map space. The minimum drop file size is an installation parameter (initially #25 blocks).

A task can also create its own drop file (see volume 2) which causes the automatic drop file to be destroyed. This can be done only if no pages have been written to the

existing drop file. The drop file is preserved for any abnormal termination and can be preserved or destroyed, at the option of the task, upon normal completion.

Write-Temporary Files

When a mass storage file with read-only access is brought into central memory, such a file can be modified; however, the modified image is paged out to the drop file and not to the read-only file. Subsequent references to the page of the read-only file access the modified page. For the duration of the job, the original source image can be referenced again only by removing the modified image from the drop file.

FILE ACCESS CONTROLS

File access is controlled by the following means.

- File security levels
- File access modes
- File ownership

No file passwords, as such, exist.

File Security Levels

Up to eight levels of security, from the lowest level (0) to the highest level (7), can be defined by an installation. The installation can define a security level at or above which the file space is patterned when the file is destroyed.

Each user is administratively assigned a maximum security level when the user number and account identifier is assigned. Whenever a LOGON is processed, the security level specified by the user is checked against the user's maximum security level. If the level specified is higher than the maximum allowed for that user, system access is denied. The same check is made whenever a user creates or accesses a file. If the security level given is greater than that allowed the user, file access is denied.

File Access Modes

The file creator can assign read, write, or read and write access permission to a file.

A file with write access can be written into by a user program or the operating system. For data files, this means that modified pages are rewritten to the original file.

An attempt to write explicitly into a read-only file produces an input/output error. An attempt to modify a page from a read-only virtual file produces a fatal error. The user can, however, map in pages of read-only files, giving those pages write temporary access.

File Ownership

File ownership determines who has access to a file. A privileged user has ownership rights over all files except local files belonging to other users. Nonprivileged users have rights that depend on ownership category. Each mass storage file is in one of the following ownership categories:

- **PRIVATE.** Private files include local files and permanent files. In the case of nonprivileged users, only the user who created the file, or only the user to whom ownership was transferred, can access a private file. Permanent files are accessible after they are attached.
- **POOL.** A group of users can access any file in a pool, as long as the pool boss who created the pool authorized those users by user number. Pool files are accessible after the pool is attached.
- **PUBLIC.** All users can access public files.

File names of files accessible to a user/suffix combination must be unique within each ownership category, but names need not be unique when all categories are considered. A user can create a private file with the same name as a public file, for instance. Two users can each have a private file with the same name, since the category of private exists for each user number. Names of private files accessible to the user must be unique, but as many as five private files with the same name might exist (one unattached permanent file and four local files, each under a different user/suffix combination).

The owner of a file has the right to establish the specifications for a file which, in turn, describe the file. These attributes are described when the file is defined or requested. The specifications can be temporary for the execution of a given task or can be permanent specifications.

File ownership categories are illustrated in figure 2-1. The control statements are described in section 4.

Private Files

A private file belongs to a specific user. Private files can be permanent files or local files. The term attached private files refers to local files and attached permanent files.

Private files are accessible only to the current owner and (for permanent files) to privileged tasks. Only the current owner has control over the files. The owner can access the file, manipulate the contents, change the file access, security level, and retention period of the file, and so forth. A nonprivileged user can give an attached private file to another user, but not to user number 000000 to which public files belong.

Each private file belongs to a particular user number and account identifier. When a private file is given by one user to another, the user number associated with the file is changed immediately. The account identifier is not changed until the file is referenced by the receiver. The system accounting tables then indicate the total time that file ownership was held by the originating account identifier.

Pool Files

A pool file is a file owned by a pool. The user defines a pool by adding the pool name to the pool list. A pool name must be one to eight characters, beginning with a letter and must be unique within the pool list.

The user who defines the pool is the pool boss. Only the pool boss can perform the following functions.

- Give pool files to another user or to the public file list.
- Purge files from the pool.
- Grant other users access to the pool.
- Remove user authorization to access the pool.
- Destroy the pool.

After a pool is defined, any user can give files to the pool. If the user gives a file to a pool to which he does not have access, he can no longer access the file because he cannot access the pool.

After a user is granted access to the pool, he can attach the pool and access any file in it. A user can attach up to four pools at one time.

Pool files are associated with a user number and all of its suffixes. When a user number is in use in interactive and batch modes at the same time, batch end-of-job procedures detach the pool from the user number if the batch job was the first to attach the pool, whether or not that pool is in use interactively. Pools first attached by an interactive user are detached when the user is no longer active under any suffix.

Public Files

Public files are accessible by the entire body of users. They contain assemblers, compilers, and other general purpose routines that augment the operating system for a particular installation. All utilities described in this manual exist as public files.

All public files belong to user number 000000, signifying ownership by the system. The security level of all public files ranges from zero through seven.

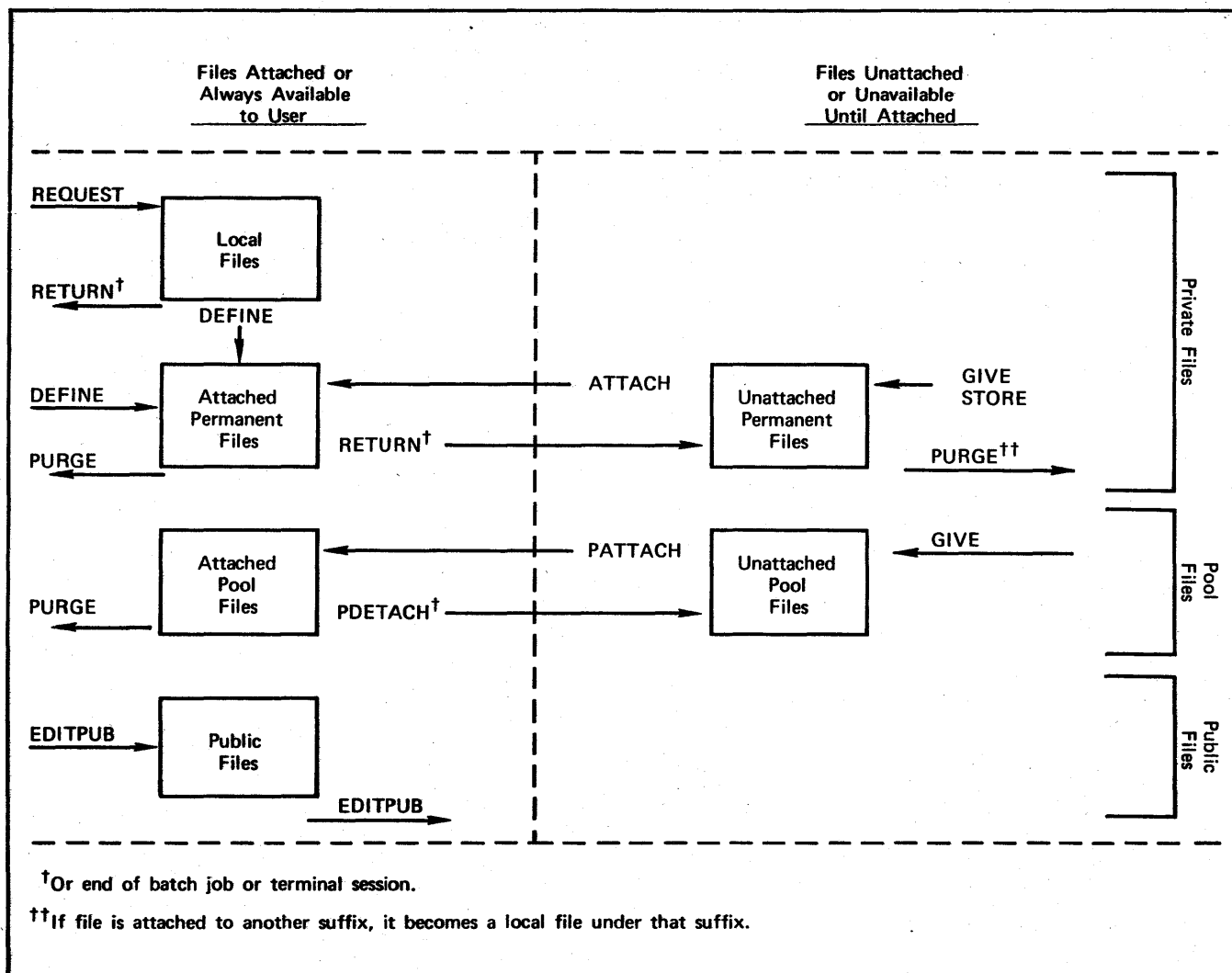


Figure 2-1. File Ownership

The list of public files is controlled by the installation administrator or privileged users.

FILE STRUCTURE

The following paragraphs describe the allocation of mass storage space to a file, the available logical record formats, and the characteristics of CYBER 200 output files.

File Space Allocation

A mass storage file can reside in up to four noncontiguous physical areas called segments. Each segment is a contiguous area; all segments are on the same disk.

One or two segments are used when the file is created; additional segments are allowed for extensions. An attempt is made to allocate the initial file size contiguously. If no space large enough is available on any disk, the file is allocated on the first unit having two segments large enough to cover the entire file. The total amount that a file may be extended is a percentage of the creation size of the file. This percentage is an installation parameter, EXTSIZ.

The segment allocation formula for extensions is:

$$S = (I * \text{EXTSIZ} / 100 - \text{ESA}) / \text{NSA}$$

S	Segment length in 512-word blocks; for large pages, S is rounded up to a multiple of 128 blocks.
I	Initial file creation length in 512-word blocks.
ESA	Extension space already allocated in 512-word blocks.
NSA	Number of remaining segments available to be allocated.

Assume that a file 64 blocks long is created contiguously and the installation parameter EXTSIZ is 50. Substituting these numbers in the segment allocation formula, the extension sizes are:

First extension:

$$10 = (64 * 50 / 100 - 0) / 3, \text{ allocate 10 blocks.}$$

Second extension:

$$11 = (64 * 50 / 100 - 10) / 2, \text{ allocate 11 blocks.}$$

Third extension:

$$11 = (64 * 50 / 100 - 21) / 1, \text{ allocate 11 blocks.}$$

If the calculated space is not contiguously available for extensions, the system allocates as much contiguous space as is available. Again assuming initial file size of 64, EXTSIZ of 50, and the largest available contiguous space of 8 blocks, the allocation of each extension becomes:

First extension:

$$10 = (64 * 50 / 100 - 0) / 3$$

10 blocks needed, but allocate only 8 blocks since that is the maximum available.

Second extension:

$$12 = (64 * 50 / 100 - 8) / 2$$

12 blocks needed but allocate only 8 blocks since that is the maximum available.

Third extension:

$$16 = (64 * 50 / 100 - 16) / 1$$

16 blocks needed but allocate only 8 blocks since that is the maximum available.

If 12 or 16 blocks become available before an extension is needed, the system allocates that amount.

If the calculated space is not contiguously available for large page extensions, the system attempts to allocate the large pages in two segments. In any large page case, the total extension space is a multiple of 128 512-word blocks.

After an extension is performed, the system sends a message to the user informing him of the extension and the new file size. If no space is available or all segments are allocated, the program is aborted with a message.

Files created for internal use by the operating system are contiguous and nonextendable. All files created by existing programs, utilities, and FORTRAN run-time, default to extendable files.

Logical Record Formats

The user can specify a file's record structure by using system interface language (SIL) calls to write the file. Section 9 describes the SIL I/O calls.

All SIL-written files have sequential file organization and fixed length blocking. The block length is dependent on the device type on which the file resides unless the user specifies a block length.

SIL supports the following four record formats.

- ANSI fixed length (F).
- Record mark delimited (R).

- Undefined structure (U).
- Control word delimited (W).

SIL sets no limit on the maximum record length that the user can specify.

The user can include two additional levels of file structure, groups and files, in R and W format files. F and U format files do not have groups or files within files.

ANSI Fixed Length (F) Record Format

SIL writes a fixed number of bytes for each F format record. It writes no delimiters or other record length information on the file. To read an F format file, the user should specify the same record length used when writing the file.

SIL uses a default record length unless the user specified a record length with the MXR parameter. The specified record length must be an integral division of the block length because F format records cannot span blocks.

SIL supplies no padding for F format records.

F format files are usually magnetic tape files containing data to be transferred between systems. The characteristics of F format files are defined in the American National Standard for Magnetic Tape Labels and File Structure for Information Interchange, X3.27-1977.

Record Mark Delimited (R) Record Format

SIL uses the R record format unless the user specifies another record format. SIL terminates each R format record with the record mark character. The user can specify a record mark character or use the default character specified by an installation parameter.

When the user transfers an R format record from the working storage area to an I/O buffer, it transfers the number of bytes specified as the working storage area length (wsl). Unless the user specifies otherwise, SIL compresses consecutive blanks. It replaces strings of more than two blanks with two character codes, ASCII ESC character (#1B) followed by the number of blanks plus #30. SIL adds #30 to the number so that the value cannot be mistaken for another ASCII control character code.

When the user transfers an R format record from the I/O buffer to the working storage area, it transfers bytes until it encounters a record mark character. If the delimiting character is not found within the number of characters specified as the maximum record length (mxr) in the file's FIT, SIL transfers the maximum record length of characters and returns an error status code. Unless the user specifies otherwise, SIL expands compressed blanks.

The user can specify any ASCII character as the record mark character. The default character is the ASCII US character (#1F). The default SIL format is, therefore, identical to the obsolete System Record Manager unstructured format. As in that format, when the US character is the record delimiter, SIL recognizes the ASCII GS (#1D) and FS (#1C) characters as group and file delimiters, respectively.

If the ASCII character RS (#1E) is the record mark character, the GS and FS characters are recognized as group and file delimiters, respectively, and US characters are ignored.

Undefined Structure (U) Record Format

A U format file has no record structure. SIL considers the file as a continuous byte string. The user specifies the number of bytes that SIL transfers on each record transfer call. The data length specified can vary with each call.

U format files are usually unstructured tape files or files containing data to be processed as byte strings.

Control Word Delimited (W) Record Format

SIL prefixes each record of data in a W format file with a word of control information (refer to figure 2-2). The control word contains the number of bytes in the record and the number of bytes between the control word and the beginning of the previous control word.

SIL can write a W format record in more than one piece. It prefixes each piece of a record with a control word describing the piece. The maximum size of a piece is 2²⁴-1 bytes.

The end-of-group and end-of-file delimiters are control words following the last record in the group or file. They are distinguished by a flag indicating the partition level of the control word.

When transferring a record to the working storage area, SIL reads the control word and then transfers the number of bytes of data specified in the control word byte count field. It does not transfer the control word to the working storage area.

When transferring a group, SIL transfers data (including record control words) until it reads an end-of-group control word. When transferring a file, SIL transfers data (including end-of-group and record control words) until it reads an end-of-file control word.

When transferring a group or file from the working storage area, SIL transfers the data and then writes the end-of-group or end-of-file control word. It enters zero in the byte count field and the number of bytes to the beginning of the previous control word in the previous size field.

When SIL transfers a record from the working storage area, it adds the control word prefix. When it transfers a group or a file, the first word of the working storage area must be a record control word. The user can include other control words within the working storage area to delimit records within the group or groups within the file.

r	p	fd	wcr	previous size	byte count
<u>Field</u>	<u>Bits</u>	<u>Contents</u>			
r	0-2	Reserved for installation use.			
	3-10	Reserved for Control Data's use.			
p	11	Parity bit used to maintain odd parity in the word.			
fd	12-13	Indicate the control word type.			
wcr	14-15	Record continuation flags.			
		00	Complete record.		
		01	First piece of record.		
		10	Middle piece of record.		
		11	Last piece of record.		
prev- ious size	16-39	Number of bytes in previous record piece including its control word.			
byte count	40-63	Number of bytes in the record, not including the control word.			

Figure 2-2. Control Word Format

Output Files

Output files contain data to be processed by an output device. They are created by task execution. When the file is closed or the task terminates normally, CYBER 200 OS gives an output file with a valid disposition code to the appropriate privileged user task for processing to the output device. After a file is processed, it is destroyed. Output files with invalid disposition codes remain in the system as unattached permanent files.

An output file is assigned the default disposition code and internal and external characteristics unless the user specifies otherwise on a ROUTE control statement or Q5ROUTE call naming the file.

Punch Files

Files punched by the unit record station are preceded by two banner cards. The first card contains the user number; the second card contains the file name as described above. Files punched in ASCII are terminated with a 6/7/8/9 separator card.

The file's external characteristic specification determines the card format punched. The available card formats are described in section 3.

Punch files cannot be grouped in families like print files.

Print Files

For the line printer, output files can be saved in families for consecutive processing. A family is Pddfammm, where dd is either decimal numbers 00 to 99 or the characters XX, and fammm is family name. Family files are held in an unprocessed state until a file name with XX as the second and third characters is encountered; then, the family is processed as a unit.

Print files undergo some processing before being printed. The following processing generates a compressed ASCII file with ANSI carriage control characters.

- Two or more blanks (#20) are compressed by replacing them with the ASCII escape control character, ECS (#1B), followed by a count of the number of blanks. The ASCII character 0 (#30) is added to the count of blanks to ensure that the result is beyond the range of ASCII control characters.
- If the internal characteristics field (ic) of the file is ASCII with ANSI carriage control, then the carriage control characters are assumed to be correct and are not looked at by the system.
- If the internal characteristics field (ic) of the file is ASCII with ASCII carriage control, then the carriage control characters are changed to ANSI by the system. The ASCII form feed control character, FF (#0C), when it occurs as the first character of the file after a unit separator, US (#1F), is replaced with the ANSI page eject control character 1 (#31). The ASCII single space, which is a line without the FF after the US, is changed by inserting the ANSI space-one-line control character, blank (#20), after the US.
- Due to file conversion, the printed file can be up to 50% longer than the original file (or family of files) up to a maximum of #1000 pages. Any file that goes over this limit is printed in parts. If a family of files goes over this limit, the family is divided at the end of a family member. If one file goes over this limit, the file is divided at a random point.
- Family files are linked together forming one file. The file separator, FS (#1C), at the end of all but the last file is replaced with a blank (#20). The start of all files is changed to Hex 31. A maximum of 101 members of a family is printed at one time. Any family that goes over this limit is printed in parts, with the family divided at the end of a family member. However, the family members that are grouped together are the first 101 members found so that they might be printed out of sequence.

Print Control Characters

Print file control characters can follow either ASCII control character conventions or ANSI conventions.

In the ASCII schema, print control is governed completely by the appearance of ASCII control characters. The FF control character must be the first character of a file or must immediately follow a unit separator (US). The ASCII control characters and their effect on vertical spacing are:

<u>Control Character</u>	<u>Vertical Spacing</u>
FF (#0C)	Page eject.
US (#1F)	Single space.

In the ANSI print control conventions, the first character of a printer/display output record is not printed or displayed; it is interpreted for vertical spacing control. The first character of each output record directed to the card punch, or any device other than a printer or a display unit, is transmitted and recorded just as any other character in the record without any special action. Characters and their effect on vertical spacing before printing or displaying the next record are:

<u>Character</u>	<u>Hexadecimal Code</u>	<u>Vertical Spacing</u>
blank	#20	Single space.
0	#30	Double space.
1	#31	Page eject.
+	#2B	No vertical advances; move to the first position of the same line.

<u>Character</u>	<u>Hexadecimal Code</u>	<u>Vertical Spacing</u>
-	#2D	Triple space.
other	other	Single space.

MAGNETIC TAPE FILES

The user assigns a tape file to a job with the REQUEST control statement (refer to section 4) or the Q5RREQUEST call (refer to section 9). The user can also access a tape via the PROGRAM statement in a FORTRAN program (refer to the CYBER 200 FORTRAN 1.5 Reference Manual). When assigning a tape, the user can specify tape characteristics including seven-track or nine-track recording, the tape density, and coded or binary format. (Nine-track tapes are always written in binary format.)

The user can also specify if the tape is labeled or unlabeled. Volume labels are processed when the tape is requested, header labels when a tape file is opened, and trailer labels when a tape file is closed. Tape label formats are given under Tape Label Processing in section 9.

The Q5RREQUEST call must be used to assign a multivolume tape file because only one volume can be specified on a REQUEST control statement.

To read or write data on a tape file, the user can issue the COPY, DUMPF, or LOADPF control statements; issue SIL calls to perform explicit I/O; or use the FORTRAN READ and WRITE statements. To use the SIL explicit I/O calls, the user must first open the file for explicit I/O using the Q5OPEN calls. The Q5OPEN call is also used to position a multifile tape volume.

The user ends the tape file assignment with the RETURN control statement or the Q5RETURN call.

A task is an executable program. (To be in executable format, a program must be compiled and processed by the LOAD utility.) A task can be called into execution by a control statement, a terminal entry, or a message from another task.

A task that initializes and starts a task is the controller of the task. The initialized task is the controllee of its controller. The controllee task can, in turn, initialize another task and thus be its controller.

The controller/controllee relationship forms a controllee chain. Messages can be sent between tasks in a controllee chain, but each task executes independently. All batch jobs are controllees of the batch processor; all interactive tasks are controllees of the interactive processor. The maximum number of tasks in a controllee chain is nine including the job controller.

To call a task into execution, the user must access the system. The user can access the system in batch mode or interactive mode. In batch mode, the user enters a job, which is a sequence of tasks to be executed. In interactive mode, the user logs in at a terminal and then enters a control statement, request line, or file name to begin execution of a task.

BATCH SYSTEM ACCESS

The CYBER 200 system processes input in batch mode when it is read from a unit record station card reader or submitted as batch input from a front-end processor. Batch input can be in the following two forms.

- A job to be executed under control of the batch processor.
- Data to be stored as an unattached private permanent file.

The user specifies the processing of the batch input on the unit record station or access station STORE card or on the link station job statement. For information on submitting batch input from the link station or access station, the user should refer to the link station or access station reference manual, respectively.

UNIT RECORD STATION BATCH INPUT

Each card deck read by the unit record station must begin with a STORE card (refer to figure 3-1) and end with a card having 6, 7, 8, and 9 multipunched in column 1. CYBER 200 OS stores the card reader input as a mass storage file. The STORE card preceding the deck specifies the name, record format, and processing of the mass storage file. The file has a security level of 0. The maximum size of the file is set by an installation parameter.

Columns	Contents
1-5	ASCII characters STORE
6	Blank.
7-12	Six-digit user number. (Add leading zeroes, if necessary, to fill the field.)
13-20	One- through eight-character account identifier.
21-28	One- through eight-character name to be given to the mass storage file into which the card deck is read (the job file). The name must not already name a public file, a permanent file belonging to the user number, or another job file currently in the system under the user number.
29	Blank.
30	Record format of the mass storage file (refer to section 2 for record format descriptions).
	R R format; #1F is the default record mark character.
	U U format (80-column binary); allowed only if column 34 is blank.
	W W format.

Figure 3-1. STORE Card Format (Sheet 1 of 2)

<u>Columns</u>	<u>Contents</u>
31-33	Blank.
34	Batch input type.
	<div>B Deck contains a batch job to be executed.</div> <div>blank Deck is not executed; it is stored as an unattached permanent file.</div>
35-48	Blank.
49	Indicates system action if a permanent file with the same name exists for the user number. If the field is blank, an installation parameter determines the system action.
	<div>C Do not create the file.</div> <div>U Purge the existing file and create a new file.</div>
50-78	Blank.
79-80	Indicates keypunch code used for deck. If the field is blank, the code is determined by an installation parameter.
	<div>26 O26 punch code.</div> <div>29 O29 punch code.</div>

Figure 3-1. STORE Card Format (Sheet 2 of 2)

SEPARATOR CARDS

Separator cards separate different parts of a batch deck. During execution of a batch job, the batch processor treats all cards between two separator cards as a separate file. For instance, a separator card is required between the control statements and a source program and between a source program and data to be processed during execution of that program.

The following are the separator cards and their uses.

7/8/9	Record separator that indicates the end of the control statements, source program, directives, or data cards.
6/7/8/9	File separator that indicates the end of the batch deck.

A keypunch conversion mode can be specified in columns 79 and 80 of a separator card. If specified, the conversion mode remains in effect for all succeeding cards until changed by another separator card. The conversion modes are the same (26 or 29) as specified on the STORE card.

Figure 3-2 shows a typical card deck containing a batch job. The job contains a control statement record, a source program record, and a data record.

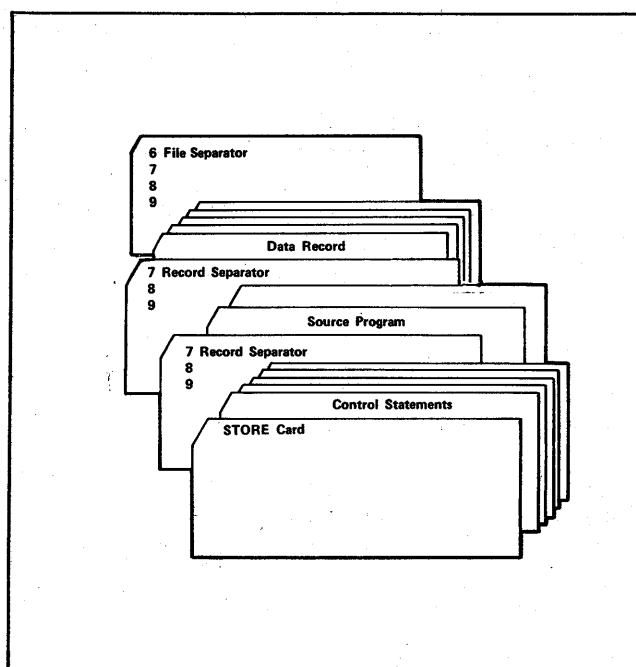


Figure 3-2. Example of Batch Deck

CARD FORMATS

A batch deck can contain either ASCII coded or 80-column binary cards. Only one card format can appear in a deck.

ASCII Coded Cards

ASCII coded cards have the punch codes listed in appendix A. Their most customary content is a source program to be compiled or assembled, data to be processed, and the control statements and directives of a batch job.

ASCII codes can correspond to either the codes of an O26 keypunch or the codes of an O29 keypunch, depending on the installation configuration. Columns 79 and 80 of a STORE card or separator card indicate whether the following ASCII coded cards are in O26 or O29 code.

80-Column Binary Cards

80-column binary cards are treated as a string of 960 bits of data corresponding to fifteen 64-bit words. All columns in a binary card are copied directly to a mass storage file with no conversion of any kind. A card in this format is also known as an absolute binary card. Bits are ordered as shown in figure 3-3.

Cards in this format must follow a card with the ASCII characters UNFORMAT in columns 1 through 8. A similar card must follow all 80-column binary cards. These UNFORMAT cards must appear in a deck in addition to any separator cards required in the deck.

	1	2	3		79	80
12	1	13	25		937	949
11	2	14	26		938	950
0	3	15	27		939	951
1	4	16	28		940	952
2	5	17	29		941	953
3	6	18	30		942	954
4	7	19	31		943	955
5	8	20	32		944	956
6	9	21	33		945	957
7	10	22	34		946	958
8	11	23	35		947	959
9	12	24	36		948	960

Figure 3-3. 80-Column Binary Format

JOB STRUCTURE

A job begins with a control statement record. Source programs to be compiled or assembled, data to be processed, or directives to be used during control statement processing can also appear in a job. Each of these elements must be preceded by a 7/8/9 separator card.

The control statement record contains one or more control statements. Within a job, control statements can be one of two types:

- Control statements that specify the name of a task to be executed as a controllee of the batch processor. The task might be one of the utilities available as a public file to all jobs or it might be the name of a controllee file produced by the control statement. These statements could also be executed through an interactive terminal.
- Special control statements that control the batch job environment. They are valid only in a batch job.

Both types of control statements are described in section 4.

The input queue manager processes the RESOURCE statement. The special control statements processed by the batch processor directly are:

COMMENT	Send message to job dayfile.
EXIT	Establish control path for abnormal job termination procedures.
READCC	Read control statements from another file.
RERUN and NORERUN	Set or reset rerun indicator for entire batch deck to determine whether the file indicated by STORE is to be destroyed or rerun in the event of system failure.

SET	Change memory limits.
TV	Define or check the error return code threshold for job termination.

Following the RESOURCE statement, the batch processor processes control statements in the order that they appear in the control statement record.

JOB PROCESSING

The following information describes the way the batch processor executes. The batch deck of figure 3-4 is presumed.

The batch input file, as such, does not have the name INPUT. Several files by the name INPUT might be created (requested) and destroyed (returned) during execution. Similarly, several files with the name OUTPUT can be created (requested) during the job's execution.

```

STORE nnnnnnaaaaaaMYJOB002      B
RESOURCE,TL=7.
TV,0+.
FORTRAN.
LOAD.
GO.
DEFINE,DATAOUT.
7/8/9 separator card

PROGRAM...INPUT...OUTPUT...DATAOUT..
.
.
.
PRINT
.
.
.
END
7/8/9 separator card

data cards

6/7/8/9 separator card

```

Figure 3-4. Example of Batch Job

When the batch deck enters the system, all cards between the STORE card and the 6/7/8/9 card are stored as an unattached permanent file with the name indicated on the STORE card. The input queue manager processes the RESOURCE statement and determines when the job is scheduled for execution. After the job is scheduled for execution, the batch processor creates a local file with the name INPUT from the first record of the file following the control statements.

It also creates a job dayfile with the name Q5DAYFLE. In the job dayfile, the batch processor, the operator, and executing tasks enter the times and events of the job, along with any status, comment, or error information produced during job execution. The batch processor also enters all operator commands related to the job, any messages a job task sends to its controller (the batch processor), and any messages the program or the operator sent to the dayfile. (A program sends messages to the dayfile with the SIL call Q5SNDMDF.)

The batch processor also creates a local file with the name Q5JOBFILE for its own use.

The batch processor then examines the first control statement (TV,0+). It recognizes it as a batch processor control statement and executes it directly, setting the error threshold value at zero.

The batch processor then examines the next control statement (FORTRAN) and determines that it is not a batch processor statement. It, therefore, presumes it to be the name of a file in executable format. The batch processor instructs the operating system to execute the file and passes any parameters on the control statement to the task. The operating system searches for a virtual code file with the name on the control statement, treating each file ownership category separately and searching in this order:

1. Attached private files for this user, including local files and attached permanent files.
2. Pool files in the order any pools have been attached by the user.
3. Public files.

Data files are ignored when searching for an executable file.

Once a file with the correct name is found, execution is initiated with the task running as a controllee of the batch processor. In this instance, assuming the user does not have an attached file with the name FORTRAN, the system finds the FORTRAN compiler available as a public file.

By default, the FORTRAN compiler assumes that the source program to be compiled resides on an attached file named INPUT. The compiler uses the file created by the batch processor and creates the local file BINARY as a result of compilation. The compilation process also creates a local file with the default name of OUTPUT to contain the FORTRAN source listing.

When the FORTRAN compiler completes execution, the batch processor examines the abort flag returned. If the abort flag is set, it initiates the job termination procedure. If the abort flag is not set, the batch processor examines the termination value returned from that task, comparing it against the threshold value of 0 established by the TV statement. If the task returns a value greater than 0 indicating errors in the compilation, the batch processor initiates the job termination procedure. The job termination procedure searches for an EXIT control statement (refer to Error Processing in this section).

If the task returns a termination value of 0, the batch processor then examines the file INPUT. Since the file has been used by the task just concluded, the batch processor returns the existing file INPUT. The batch processor then creates a new local file named INPUT from the next record that follows the control statements.

Notice that as a result of the creation, destruction, and re-creation of the file INPUT, it is not possible for more than one task to reference the same record in the batch deck.

The file OUTPUT has its name changed by the batch processor to a name that indicates a member of a family of print files. OUTPUT becomes P00hmmss, where hmmss is the time the batch processor is started. The time uniquely identifies the family of print files.

The batch processor then examines the next control statement and either takes the action directed or starts a file with the name on the statement, as before. In this example, the file LOAD is found as a public file. LOAD executes as a controllee of the batch processor.

During execution, LOAD uses a file with the default name BINARY as its input. It attempts to satisfy external references from the default file SYSLIB. It creates a local file of executable code that has the default file name GO. It also creates a file with the default name OUTPUT to contain the load map. At the end of LOAD execution, the batch processor again checks the code returned by the task against the threshold value of 0, examines the file INPUT, and searches for a file with the name OUTPUT. Since the file INPUT has not been used during LOAD execution, it remains as it was created. The name of the file OUTPUT is changed to the name P01hmmss. The batch processor then examines the next control statement.

In searching for a file with the name on the control statement GO, the system finds the local file GO created by LOAD. GO then executes as a controllee.

During execution of the FORTRAN program, the program is assumed for this example to read from file INPUT and write to the file OUTPUT. Assume it also writes to a file the PROGRAM statement equated with the local file name DATAOUT.

At the end of execution of the task GO, the batch processor finds the file INPUT has been used and returns it. No more records exist to be used to create a file INPUT. The name of OUTPUT is changed to P02hmmss.

The next control statement is DEFINE, which makes the local file DATAOUT a permanent file. The file DATAOUT therefore remains in the system; all other files for the job are destroyed at job termination.

The batch processor then ends the job. As part of job ending procedures, the batch processor changes the name of the dayfile from Q5DAYFLE to PXXhmmss. Adding a file with a name beginning with PXX completes the family of print files, and the family is routed to a line printer. Family files are printed as one file.

The file MYJOB002 is then destroyed.

INTERACTIVE SYSTEM ACCESS

The user can access the CYBER 200 OS through a terminal logged into the front-end processor operating system (NOS or NOS/BE). The appropriate entries used to establish interactive communication with the CYBER 200 are described in the Link Reference Manual and the Access Station Reference Manual.

Interactive access begins with a LOGON command that identifies the terminal user for security and accounting purposes. Access ends with a BYE request line.

The format of the LOGON command is shown in figure 3-5. Blanks separate fields in the line. A carriage return terminates the entry.

LOGON usern suffix account level		
userno	User number of six digits that uniquely identifies the terminal user. Leading zeros should be specified if needed for six digits.	
suffix	Letter A, B, C, or D under which a subsequent task is to execute.	
account	Account identifier of one through eight characters.	
level	Single character defining a security access level. Optional.	
	Undefined	Level 0
	P or omitted	Level 2
	A	Level 3
	S	Level 5
	K	Level 7

Figure 3-5. LOGON Format

The LOGON command establishes a user number and a suffix for execution of a task. The user number establishes the files that can be accessed.

An interactive task can be initiated under any of the four available suffixes, A, B, C, and D. A user can log on to the system under a given user number and suffix, initiate a task, execute a BYE request line, and then log on again with a different suffix to continue operations while the first task is executing. The user can also change the suffix with a request line as an alternative to another logon.

If a user has batch jobs and interactive tasks executing at the same time, actions of the batch jobs should be considered before interactive tasks are initiated because permanent file name conflicts might occur. All batch jobs execute under suffix D, which restricts interactive operations to suffixes A, B, and C under the same user number while the batch job is executing. If a batch job is active, all interactive entries under suffix D are ignored, except request lines.

Four types of entries can be made from an interactive terminal after LOGON:

- A request line that makes a direct request to the operating system. These entries begin with a special character defined by the installation.

- A file name that calls a private file or pool file into execution. The file or pool must be attached and must have controllee file format prepared through the loader.
- A control statement that calls a public file into execution.
- A message to be sent to an executing task.

REQUEST LINES

A request line makes a direct request of the operating system. Each request line is prefaced by a special character that distinguishes it from other types of entries from a terminal. The special character, by default, is the character \$, but it might be changed by the installation.

Possible requests are as follows. The special character is indicated by \$.

\$T	Get current date and time in format mm/dd followed by hh.mm.ss.
\$S	Get current state of program active under user's suffix; possible responses appear in table 3-1.
\$BB	List current accounting information for program active under user's suffix. Remaining time units are displayed.
\$?	Get current date, time, accounting information, and program state for program active under user's suffix; equivalent to \$T, \$BB, and \$S.
\$SU	List current activity of programs active under all user's suffixes, A, B, C, and D.
\$PR	List number of job tasks in interactive (I) class waiting for execution.
\$P	List attached pools for this user.
\$U	List time consumed by user since logon.
\$BP	List time remaining in repository to which the user belongs. Time remaining reflects balance after initial time increment is granted at logon, and after any additional time is drawn from the repository. Refer to Accounting in this section.
\$G+xx	Draw xx minutes from the repository. Response might indicate no pool to be referenced.
\$G-xx	Return xx minutes to the repository.

\$I Send program to current interrupt routine, if program is so enabled (refer to Message Interrupts in this section).

\$OP message Send message to operator's terminal. When the request has been completed, the system responds with OK.

\$suf Disconnect terminal interface with current suffix and remain active under new suffix suf.

\$BYE End interactive access to CYBER 200.

TABLE 3-1. PROGRAM STATES

Response	Meaning
RUNNING	In execution.
WAIT ALT	Waiting for CPU assignment.
WAIT TPE	Waiting for tape assignment.
WRT CNTR	Waiting for controller to be assigned initial memory resources.
WRT CNTE	Waiting for controllee to be assigned initial memory resources.
RCV CNTR	Waiting for message from controller.
RCV CNTE	Waiting for message from controllee.
RCV PDP	Reserved for Control Data Corporation.
SND CNTR	Waiting to send message to controller.
SND CNTE	Waiting to send message to controllee.
SND PDP	Reserved for Control Data.
SND OPR	Waiting to send message to the operator.
SND TTY	Waiting to send message to the teletype.
DUMPING	Input/output being dumped to disk.
FINISH	Finished; clean-up is in progress.
SUSPEND	Suspended.
WAIT MP	Waiting for minus page to be assigned.
RCV OPER	Waiting to receive message from the operator.
WAIT mfx	Waiting for mainframe identified.
NIL	No tasks in execution.

Any tasks active when either the change suffix request line or the BYE request line is entered remain active and continue to the end of execution.

A special BREAK character can be used to terminate the task currently running. The character aborts the current controllee and transfers control to its immediate controller. If the aborted task has no controller, it can be restarted by executing its drop file. The default BREAK character is !, but it might be changed by the installation.

PRIVATE FILE EXECUTION

Any attached private file in executable format can be called into execution through an entry that has the general format shown in figure 3-6.

taskname / TL=t,PRIORITY=p,WS=w,LP=lp / string	
or	
taskname string / TL=t,PRIORITY=p,WS=w,LP=lp	
taskname	Name of task to be placed into execution (one through eight letters or digits). The first six characters must not duplicate those of any other file to be called into execution.
TL=t	Task time limit in system seconds [†] (decimal integer between 1 and 599 940).
	If TL=t is omitted, the task time limit is 60 system seconds.
PRIORITY=p	Task priority, 1 (lowest) to 15 (highest). If the specified priority exceeds the maximum priority the installation specified for interactive tasks, the task priority is set equal to the maximum priority.
	If PRIORITY=p is omitted, the job priority is the installation-specified default priority for interactive tasks.
WS=w	Maximum working set size in blocks (decimal integer). If the specified limit exceeds the maximum limit the installation specified for interactive tasks, the task is aborted.
	If WS=* is specified, the maximum working set size is all of allocatable memory.
	If WS=w is omitted, the maximum working set size is determined by an installation parameter.

Figure 3-6. Interactive Task Call Format
(Sheet 1 of 2)

[†] A system second is one million STUs. If desired, an installation can substitute SBUs for system seconds as the time limit unit. The calculation of an STU or an SBU is described in volume 2.

LP=lp	Maximum number of large pages that can be assigned to the task (decimal integer). If the specified limit exceeds the maximum limit the installation specified for interactive tasks, the task is aborted.
	If the large page limit, when multiplied by 128, exceeds the working set size limit, the task is aborted.
	If LP=lp is omitted, the large page limit is zero.
string	Character string to be passed to the task. The format of the character string depends on the coding of the task. The string is delimited by blanks. The delimiting blanks are not passed to the task.

Figure 3-6. Interactive Task Call Format
(Sheet 2 of 2)

Only the task name is required in the execute line. A string to be passed to the task can be entered before or after the resource parameters. If resource parameters are specified, a slash must precede the parameters and a second slash must separate the parameters from the string if it follows the parameters. All parameters specified should conform to the conventions used on system-supplied control statements. All addresses are assumed to be hexadecimal values; any other number is assumed to be a decimal value unless preceded by #.

CONTROL STATEMENT EXECUTION

All control statements described in section 4, except the batch processor control statements, can be entered through the terminal. The format for interactive and batch use is the same, except where differences are specifically noted.

Many of the utilities can be called by a complete control statement or by utility name alone. When only the utility name is entered, the utility responds by sending prompting messages to the terminal. In response to the prompting message, the terminal user should enter an option, terminating each entry with a carriage return. Additional prompting messages can appear.

Figure 3-7 illustrates an interactive call to the loader. © indicates the terminal key that produces a carriage return and line feed (usually marked NEW-LINE). Interactive use of the loader requires special care. The terminal user must type space © to indicate no options or when terminating options. The terminal user begins by entering:

LOAD ©

System response is shown in lowercase letters; uppercase letters indicate the terminal user reply to load files XA, XB, and XC and have the loader write the executable virtual code file to TONY with a load map on file PRINTMAP.

XB, and XC and have the loader write the executable virtual code file to TONY with a load map on file PRINTMAP.

input ?	Request from loader.
XA, XB, XC ©	User enters names of files containing modules output by a compiler.
origin ?	Request from loader.
28000 ©	User enters bit address where first module is to be loaded.
entry ?	Request from loader.
©	User indicates no option.
any other options ?	Request from loader.
CN=TONY ©	User indicates controllee option to create virtual code file TONY.
continue	Answer from loader.
OU=PRINTMAP ©	User indicates load map file name.
continue	Answer from loader.
Δ ©	User terminates options and calls for start of load operations.

Figure 3-7. Example of Interactive LOAD Call

MESSAGE INTERRUPTS

Tasks can be programmed such that they expect messages from the terminal (refer to section 8). Messages can have any format. No buffer exists for terminal entries that would allow several messages to be entered before the first is accepted by a task. A second message for an executing task should not be entered until the first is accepted; otherwise, the second might overwrite the first. Programming the task to prompt for input and to acknowledge output can regulate message flow. Messages can be sent to a task only when the task is executing under the suffix currently in use.

ACCOUNTING

The operating system provides a set of features that an installation-supplied accounting routine or the operator can use to control system resources consumed by an individual user or a group of users.

The accounting system can be used as follows:

1. A user number is associated with a division and a repository at the time it is authorized system access. In order to access the system, the user must withdraw an allocation of system resources from the repository before he can use the system under his user number. These functions are performed by the installation-provided software. The system billing unit algorithm is installation defined. The default unit is time in microseconds.
2. Having gained access to the system, a user can execute tasks and jobs. Statistics are accumulated over each task or batch job in both the cumulative accounting buffer and the accounting file. The installation has the option to use the statistics in the buffer to debit the user's allocation of resources withdrawn from the repository. It can also use the equivalent statistics in the accounting file to charge the user.

JOB SCHEDULING

Each batch input file entered in the system is processed by the input queue manager. It assigns each batch job a job selection number that determines its position in the input queue. The job selection number is based on the job priority and on the time the job entered the system. The batch user can specify a priority on the RESOURCE statement as described in section 4. Jobs with the same priority are positioned in the queue according to the time they entered the system; older jobs have a higher job selection number.

Upon termination of an executing job or task, the input queue manager determines the next job to give to the CPU scheduler. Starting with the job with the highest job selection number, the input queue manager selects the first job in the queue that meets scheduling constraints. A job may be bypassed for any of the following reasons:

- Jobs for its job category are not being accepted at this time (the job category is turned off).
- The maximum jobs for the job category are already executing.
- The user already has a batch job executing.
- A requested resource limit exceeds the maximum limit for the category.
- Reservation of the requested working set size would overcommit memory beyond the allowed overcommitment percentage.
- The time limit for the job, when added to the time limits of all other executing jobs, would exceed the maximum rerun time specified by the installation.

The input queue manager can bypass a job in the queue up to an installation-specified limit; a job with a bypass count equal to the bypass limit is scheduled before any following job in the queue. An exception are jobs being held in the

queue because of excessive resource requests (refer to Resource Allocation). These jobs remain in the queue until the operator increases the allowed limits or evicts the job.

Interactive tasks go directly to the CPU scheduler without processing by the input queue manager.

RESOURCE ALLOCATION

The maximum system resources allowed a job or task depends on the job category to which it belongs. All interactive tasks belong to the INTRACTV category. Batch jobs belong either to the default category, JDEFAULT, or to a job category defined by the installation. The job's RESOURCE statement determines its job category.

A job category is defined by the following installation specifications:

- The 1- to 8-character mnemonic that identifies the category
- The maximum number of jobs belonging to the category that can concurrently execute.
- The following limits for each job in the category:
 - Maximum and default priority
 - Maximum time limit
 - Maximum working set size
 - Large page limit

When a batch job enters the system, the input queue manager determines if the user is validated for the job category specified on the RESOURCE statement. (All users are validated for the JDEFAULT category.) It then checks that the resource limits requested on the RESOURCE statement do not exceed the machine memory limits. If a requested limit exceeds the machine limit, the job is aborted.

The input queue manager then checks if the requested limits are within the maximum limits set for the job category. If the job priority exceeds the job category priority limit, the job priority is set at the maximum for the job category. If the requested time limit, working set size limit, or large page limit exceeds the respective job category limits, the job is held in the input queue until the operator enables its execution or evicts the job.

If the requested limits do not exceed the job category limits, they become the initial limits for the job.

Similarly, an interactive task is aborted if a memory or time limit requested on its execute line exceeds the limit for interactive tasks. If its requested priority exceeds the maximum priority for interactive tasks, its priority is set at the maximum. Otherwise, its requested limits become its initial limits.

Within a job, the user can change its working set size limit and large page limit with a SET statement. Within a task, the user can change its large page limit with the Q5SETLP call and its time limit with the Q5SETTL call.

ERROR PROCESSING

When the batch processor begins execution of a job, it sets the initial threshold value to the installation-defined threshold value. The threshold value is the maximum return code that a task can return without job termination. If an error occurs during execution, system utilities return one of the following codes.

ERROR	Nonfatal errors occurred (termination value 4).
FATAL	A fatal error occurred (termination value 8).

The user can set the job threshold value with the TV control statement (refer to section 4).

The batch processor initiates termination procedures if a task returns either of the following conditions.

- The abort flag is set.
- The task termination value is greater than the threshold value and abnormal termination control is not enabled.

To terminate the job, the batch processor searches subsequent control statements for an EXIT statement (refer to section 4). If it finds an EXIT statement, it resumes job execution with the control statement following the EXIT statement. If it does not find an EXIT statement, it terminates the job immediately.

ABNORMAL TERMINATION CONTROL

The abnormal termination control (ATC) feature allows the user to set up interrupt processing if the system fails during program execution. The system failure may or may not be caused by the user's program. (The errors are listed in table 3-2.) The user can process computation errors using the FORTRAN Library Data Flag Branch Manager routines described in the CYBER 200 FORTRAN 1.5 Reference Manual.

ATC Interrupt Subroutine

To set up interrupt processing, the user must write an interrupt subroutine to perform the error processing that the program requires. The interrupt subroutine could test the error code to determine if the program can continue. It could also print the contents of the program variables at the time the error occurred to assist in analysis of the error. The first line of the subroutine must have the format shown in figure 3-8. The system error codes passed to the subroutine are listed in table 3-2.

The user can include a Q5RFI call in the interrupt routine to return control to or to abort the interrupted task. If the user omits the Q5RFI call, the task is aborted when the interrupt subroutine terminates.

SUBROUTINE subname(errcode,pcounter,invis,regs)

or

ENTRY subname(errcode,pcounter,invis,regs)

errcode System error code (refer to table 3-2).

pcounter Virtual bit address where the system detected the error (contents of program counter).

invis Invisible package of interrupted task (40-word array).

regs Register file of interrupted task (256-word array).

Figure 3-8. Interrupt Subroutine Header

TABLE 3-2. SYSTEM ERROR CODES

Hexadecimal Code	Meaning
5	The instruction is not in the CYBER 200 instruction set.
6	The exit force instruction does not have a pointer to a system message to be executed.
7	Illegal request.
8	Parity error in data transfer between the CPU and central memory.
9	Job unrecoverable due to an outstanding I/O request.
A	A C50x request did not contain a file segment table ordinal.
B	Illegal C504 request.
25	The drop file page size differs from the page size used on the currently executing system.
28	A write violation occurred while the system was swapping in a page referenced by the job.
29	The job referenced a page within the virtual system address range.
2A	The drop file map is full; the job can define no more virtual regions.
2B	The job class of the job is not allowed use of large pages.
2C	The job referenced a page in the library reserved area.
2D	Drop file overflow; no more virtual space can be mapped into the drop file.
2E	The drop file map is full; no more virtual space can be mapped into the drop file.
2F	A virtual system call caused drop file overflow.
30	Time limit; the system allocates time for processing the interrupt subroutine.
31	The WRPLY routine received an I/O error.
40	Bound implicit map anomaly.
51	The file segment table is full.
FF	A disk error occurred during paging.
209	No source file.
210	No drop file.
212	The pointer to the system message Alpha was zero.
213	The pointer to the system message Alpha was out of bounds.
215	No error exit address was specified and the system message encountered an error.

The following system messages are forbidden in an ATC interrupt subroutine. (System messages are described in volume 2 of the CYBER 200 OS Reference Manual.)

- Explicit I/O system message that uses an interrupt subroutine (f=0050, c=5).
- Give up CPU system message (f=0052) that waits for completion of an I/O request issued before the ATC interrupt subroutine is entered.

These system messages can cause an interrupt deadlock with ATC, preventing completion of the job. Explicit I/O requests issued as a result of a FORTRAN statement or an SIL call within the ATC interrupt subroutine are processed correctly.

Terminal interrupts are ignored within the ATC interrupt subroutine even if the subroutine contains an SIL call or system message to process terminal interrupts. If the ATC interrupt subroutine returns control to the interrupted task, the task can then process terminal interrupts, although it may not process correctly the terminal interrupts received during ATC processing.

Enabling and Disabling ATC

The user inserts a Q5ENATI call in the program where abnormal termination control is to begin. On the call, the user specifies the interrupt routine to be used. To change the interrupt subroutine used, the user issues another Q5ENATI call naming another subroutine.

The user inserts a Q5DISATI call in the program where abnormal termination control is to end.

Abnormal termination control does not function under any of the following conditions.

- The program is already in interrupt mode.
- The program has exceeded its error recovery limit.
- The program encounters a second time limit error.

Abnormal termination control does not function if the program is already in interrupt mode. The program is in interrupt mode when it is processing a terminal interrupt, when it is performing certain I/O functions, or when it is in the abnormal termination control interrupt subroutine. If a fatal error occurs while the program is in interrupt mode, the program aborts without abnormal termination control processing.

Abnormal termination control does not function if the program has exceeded its error recovery limit. The user can specify an error recovery limit (1 to 256 recoveries) on the Q5ENATI call. If he does not specify an error recovery limit, the default limit of 25 recoveries is used.

Abnormal termination control does not function when the program encounters a second time limit error. After encountering the first time limit, the program is allocated additional time for interrupt subroutine processing. (The amount of time is set by an installation parameter; it is usually 500 000 STUs.)

Control statements are executed within a batch job or through an interactive terminal. Table 4-1 lists all control statements by general function.

All control statements described in this section can be executed interactively except the COMMENT, EXIT, READCC, RERUN, NORERUN, RESOURCE, SET, and TV control statements. Except for RESOURCE, these control statements are processed directly by the batch processor. RESOURCE is processed by the input queue manager. A user file with a name matching one of these control statement names cannot be executed from within a batch job.

The functions performed by the utilities described in this section are the same for both batch and interactive system access.

The format of parameters passed to the utility generally follows the same conventions:

- Unless indicated otherwise, all addresses are assumed to be hexadecimal constants.
- All other digit strings are assumed to be decimal digits unless preceded by the character # indicating hexadecimal digits.

For parameters other than addresses, decimal and hexadecimal can usually, but not always, be substituted for one another. Substitution cannot occur where decimal or hexadecimal is specifically noted in the parameter descriptions.

BATCH JOB CONTROL STATEMENT FORMAT

All control statements submitted as part of a batch job have the same general format:

task,parameters. comment

Any blanks before the task name are ignored. The task name can be followed by any of the following separator characters, although the comma is shown as the separator in all formats in this manual.

(, blank

If the task name and a parameter are separated by more than one blank, only one blank is passed to the task.

A control statement must be terminated by either a right parenthesis or a period. Blanks to the right of the terminator are ignored. If a terminator does not exist on a card, the card immediately following is presumed to be a continuation. (A COMMENT control statement cannot be continued.) No special continuation character exists for batch control statements.

Any characters after the terminator are presumed to be a comment. These characters are copied to the dayfile, but are not otherwise processed.

The parameters of a control statement are checked by the utility called, not by the batch processor itself. Any errors in the parameters submitted or any errors encountered during execution of the utility are reported to the dayfile unless otherwise noted. Successful execution is also reported with a status message to the dayfile.

INTERACTIVE UTILITY EXECUTION

The syntax of a control statement entered through the terminal can take the form of either:

- Only the utility name, with an optional right parenthesis or period terminator.
- A complete control statement with all parameters on one or more lines.

When only the utility name is entered, the utility responds with a prompting message, such as PLEASE SPECIFY PARAMETERS. The prompting message might also include more specific information about appropriate entries, such as a message SPECIFY: FILENAME, LENGTH, OPTIONS. In response, the user should comply with an entry of one parameter or a string of parameters separated by commas. Each entry must be terminated by a carriage return.

When the control statement is entered on a single line, the task name must be followed by a blank, a comma, or a left parenthesis. Other parameters can be separated by blanks or commas also. Depending on the utility, some parameters have subfields separated by the character slash. Any blank immediately adjacent to a parenthesis, comma, slash, or period is ignored.

With the exception of the LOAD utility (refer to figure 3-7), the control statement can be entered on more than one line. (No prompting occurs between continued lines.) To continue a control statement entered interactively, the character & must be the last character before the carriage return. Thus the next entry line is presumed to be a continuation of the string of characters in the previous entry. Several lines can be concatenated up to a limit of 4096 characters. The following entries are equivalent:

RETURN FILE1,FILE2,FILE3,FILE4

RETURN FILE1,FILE2,FI&
LE3,FILE4

Any error in the parameter submitted, or any errors encountered during execution of the utility are reported at the terminal. Successful execution is also reported with a status message.

TABLE 4-1. CONTROL STATEMENT FUNCTIONS

Name	Function	Name	Function
Batch Job Only		GIVE	Change file owner.
COMMENT	Send message to dayfile.	LOADPF	Reload files.
EXIT	Set abnormal termination path.	PURGE	Evict permanent or pool files.
NORERUN	Set norerun status.	REQUEST	Create local file or assign tape file.
READCC	Read alternate control card file.	RETURN	Ends file assignment.
RERUN	Set rerun status.	Pool file	Create, access, or destroy a pool of files that can be accessed by other users.
RESOURCE	Set job limits.	ROUTE	Specify file disposition.
SET	Change memory limits.	SWITCH	Change file characteristics.
TV	Set threshold value.		
System Access		Debugging	
STORE	Establish batch access through a card reader (refer to section 3).	DEBUG	Symbolic debug (refer to section 6).
LOGON	Establish interactive access through a terminal (refer to section 3).	DUMP	Dump drop file (refer to section 6).
		LOOK	Symbolic dump (refer to section 6).
File Management		File Update	
ATTACH	Attach permanent files.	UPDATE	Maintain card image file (refer to section 5).
AUDIT	List file information.		
COMPARE	Compare file contents.	Privileged User Only	
COPY	Copy file.	EDITPUB	Add or destroy public file.
DEFINE	Create permanent file or make local file permanent.	Load File	
DUMPF	Dump files.	LOAD	Create controllee file.
FILES	List files.	OLE	Edit object library.

ATTACH - ATTACH PERMANENT FILES

The ATTACH control statement (refer to figure 4-1) accesses an unattached permanent file. The file is attached to the user number/suffix combination under which the job is executing. A permanent file can be attached to only one suffix at a time. Local file names and attached permanent file names must all be unique for a particular user/suffix combination. File attributes are not changed with the ATTACH control statement, and no message is produced upon successful completion.

The user can specify the WAIT parameter on the ATTACH statement so that if a file is attached to another suffix, ATTACH waits until the file is no longer attached and then attaches the file. The site sets the length of time that ATTACH waits with an installation parameter.

If the ATTACH control statement specifies a list of permanent files to be attached, all files that can be attached, even if some files in the list cannot be attached. If a permanent file is already attached, the error is nonfatal and no message is produced. The step terminates with a fatal error if no permanent file lfn exists or if a local file lfn exists at the suffix.

If the permanent file lfn is already attached to a different suffix, the WAIT parameter determines processing. If WAIT=YES is specified, ATTACH waits until the file is no longer attached. If the file does not become available in the length of time defined by the installation or if WAIT=NO is specified, the step terminates with a fatal error. If the ATTACH statement is issued interactively, an informative message is displayed while waiting for the file.

$\text{ATTACH, } \left\{ \begin{array}{l} \text{lfn-list} \\ * \end{array} \right\}, \text{WAIT=x.}$	
lfn-list	Parameter list of 1 through 16 file names specifying existing permanent files to attach to this job. Each file name must be one through eight letters or digits beginning with a letter (except drop file names).
*	Indicates the system should attach all permanent files belonging to this user.
WAIT=x	Indicates the system action taken if a file to be attached is already attached to another suffix.
	Y Wait until the file is unattached and then attach the file.
	N Abort the task immediately.

Figure 4-1. ATTACH Control Statement Format

AUDIT - LIST FILE INFORMATION

The AUDIT control statement lists information relating to the file status of permanent, public, or pool files. Local files cannot be audited. Permanent files need not be attached. Files to be audited can be selected by file name, pack residence, or pool name. Further qualification of files to be audited can be specified by date, time, and type of the last file operation. Attached or unattached status is not reported on the output file.

A nonprivileged user can only audit public files, permanent files belonging to that user, and files in pools that the user is authorized to attach. A privileged user can audit all public, pool, or permanent files in the system.

Figure 4-2 shows the AUDIT control statement format. All parameters are optional and can appear in any order.

Parameters work in logical combinations to determine files to be audited. When PN, UN, PF, and POOL are all omitted, only private permanent files associated with the user are audited. When more than one of these parameters is specified, files must meet all criteria specified before they are audited. Similarly, the options specified by the OP parameter operate in combination to select files. A parameter OP=CM, for example, selects files created or modified after a specified date and time; a parameter NCM selects files that have not been created and have not been modified after a specified date and time. The UN and POOL parameters interact as shown in table 4-2. A nonprivileged user cannot specify any user numbers in UN=list form except 0 (for public files) and the user number under which the task is to execute.

AUDIT,PN=pkid-list,PF=lfn-list,UN=userno, POOL=pl-list,OP=opts,DT=mmddyy,TM=hhmm, LO=x,OU=lfn/len/dc.	
PN=pkid-list	List of 1 through 16 identifiers of packs, separated by commas, to be searched for files satisfying the other parameters. If the PN parameter is omitted, all active packs are searched.
PF=lfn-list	List of 1 through 128 names, separated by commas, of files to be audited.
UN=userno	Indicates files to be audited. For a nonprivileged user either or both of the following can be specified: 0 All public files. userno User number under which AUDIT is executing. Default.

Figure 4-2. AUDIT Control Statement Format
(Sheet 1 of 2)

For a privileged user:		If the OP parameter is omitted, default is none of these options.	
u-list	List of 1 through 128 user numbers, separated by commas, of files to be audited. User number of zero specifies public files.	DT=mmddyy	Date to modify the A, C, or M option, in format indicating month of year, day of month, and the last two digits of the year.
ALL	Indicates that all files are to be audited.	If the DT parameter is omitted, default is the current date.	
<u>POOL</u> =pl-list	List of 1 through 128 pools, separated by commas, to which user is attached.	TM=hhmm	Time to modify the A, C, or M option, in a format indicating hours and minutes in a 24-hour clock.
The PATTACH control statement must precede use of this parameter.		If the TM parameter is omitted, default is 0000, which is midnight preceding the day specified by the DT parameter.	
OP=opts	File characteristics which qualify files selected by UN, PN, POOL, or PF parameters.	LO=x	Indicator of type of audit:
Options A, C, M, N and X can be specified in any order; commas must not appear between these characters.		F Full audit.	
A Files accessed since date and time specified.		P Partial audit. Default.	
C Files created since date and time specified.		OU=lfm/len/dc	
M Files modified since date and time specified.		File to which audit information is to be written:	
N Reverse the meaning of any A, C, or M specified. That is, the appearance of N changes the meaning of A from accessed to not accessed, the meaning of C from created to not created, and the meaning of M from modified to not modified.		lfm	Name of file. Must be 1 through 8 letters or digits beginning with a letter. Default is OUTPUT.
X Files expired. That is, files whose creation date plus retention period specifies a date preceding the current date.		len	Number of small pages in file. When /len is omitted, the default is #40.
		dc	Disposition code indicating processing of file:
		PR	Print on any available printer at the end of the utility.
		When /dc is omitted, the utility does not print the file.	

Figure 4-2. AUDIT Control Statement Format (Sheet 2 of 2)

TABLE 4-2. INTERACTION OF UN AND POOL PARAMETERS FOR AUDIT, DUMPF, AND LOADPF

Files Processed	Privileged User						Nonprivileged User					
	No UN		UN=list		UN=ALL		No UN		UN=list		UN=ALL	
	No POOL	POOL=list	No POOL	POOL=list	No POOL	POOL=list	No POOL	POOL=list	No POOL	POOL=list	No POOL	POOL=list
User private files	X						X		X	X	X	X
Listed user private files (and public files if UN=0)			X	X					†	†		
Listed pool files		X		X				X		X		X
All files regardless of owner (including public and pool files)					X	X						

† Only AUDIT allows listing of the user's private files and the system public files.

Information produced by a partial audit is listed in figure 4-3; a partial audit ends with the DLEN column. If a file exists on storage as more than a single segment, separate lines appear for each segment. Dates appear as month, day, and year; time appears as a 24-hour clock. All values are decimal unless preceded by #.

Figure 4-3 shows an example of audit output produced by the control statement:

AUDIT,UN=0,LO=P.

The following are the column headings used in a full AUDIT listing and the information given under each heading.

Heading Meaning

NAME File name.

OWNER Owner user number, public (0) or pool name.

Heading

TYP File type: virtual code (VC) or physical data (PD).

BT Blocking type: character count (C) or non-SIL file (blank).

RT Record type: ANSI fixed length (F), record mark delimited (R), undefined (U), or control word (W).

FC File category: batch file (B), user file (U), system-generated drop file (S), or not defined (N).

ACS Access permission: read (R) and/or write (W).

EXT File allocation: segmentable (S) and/or extendable (X).

NAME	OWNER	TYP	FC	RT	BT	ACS	EXT	SL	PACKID	UN	SADDR	SLEN	DLEN
LABEL02	0	PD	U	W	C	R	SX	0	TPAK02	3	#00140	1	
PFI02	0	PD	U	W	C	RW	SX	0	TPAK02	3	#00141	31	
BADS9102	0	PD	U	W	C	RW	SX	0	TPAK02	3	#0FF00	1	
T9408	0	VC	U	R	C	RW	SX	2	TPAK04	4	#05944	81	0
XGIVE	0	VC	U	W	C	RW	SX	2	TPAK04	4	#04682	58	0
DIMPLES	0	VC	U	R	C	RW	SX	2	TPAK04	4	#033AF	183	0
	0	PD	U	W	C	NO	SX	0	TPAK04	4	#00000	0	

Figure 4-3. AUDIT Sample Output

Heading	Meaning
SL	Security level.
PACKID	Pack identifier of mass storage file.
UN	Logical unit number of device on which pack resides.
SADDR	Hexadecimal physical sector address of segment.
SLEN	Number of small pages in segment.
DLEN	Number of small pages in drop file associated with a virtual code file.
FACT	Accounting information.
DORG	Creation date (that is, date of origin).
TORG	Creation time (that is, time of origin).
DOLA	Date of last file access.
TLR	Time of last file access.
DOLM	Date of last file modification.
TOLM	Time of last file modification.
EXP	Expiration date (that is, creation date plus retention period).

COMMENT - SEND MESSAGE TO DAYFILE

The COMMENT control statement is valid only within a batch job as it is executed directly by the batch processor. COMMENT causes the accompanying character string to be inserted in the dayfile.

Figure 4-4 shows the COMMENT control statement format. No space need appear after the required period; no ending punctuation is needed at the end of the message. Multiple COMMENT control statements are required to send a message longer than the number of columns available on a single card or card image.

COMMENT.message

message Characters to be sent to the dayfile. Any characters can be specified, but only those available on the line printer should be specified.

Figure 4-4. COMMENT Control Statement Format

COMPARE - COMPARE FILE CONTENTS

The COMPARE control statement compares the contents of one attached file with the contents of another. If contents do not compare, nonmatching words are written to the dayfile of a batch job or are displayed at an interactive terminal. Both physical and virtual files can be compared.

The files compared must have the same record format. COMPARE considers each file as a continuous bit stream, not as a sequence of records.

To compare a tape file, the user must first assign the tape file to the job with a REQUEST control statement. COMPARE cannot automatically switch between volumes of a multivolume tape file.

Figure 4-5 shows the COMPARE control statement format. The first two parameters are required. All other parameters are optional and can appear in any order.

COMPARE,alfn,blfn,L=len,A=aadr,B=badr,N=lt.

alfn,blfn Names of files to be compared.

L=len Hexadecimal number of words to be compared.

If the L parameter is omitted, comparison stops at the end of the shorter file.

A=aadr Relative hexadecimal word address in file alfn at which comparison is to begin, counting the first word of the file as 0.

If the A parameter is omitted, comparison begins with the first word of file alfn.

B=badr Relative hexadecimal word address in file blfn at which comparison is to begin, counting the first word of the file as 0.

If the B parameter is omitted, comparison begins with the first word of file blfn.

N=lt Decimal number of nonmatching words allowed before comparison stops. Both the nonmatching words and their relative locations are displayed.

If the N parameter is omitted, default is 1.

Figure 4-5. COMPARE Control Statement Format

Any compare operation for virtual files should take into consideration that the first 512 words of a virtual file contain the minus page and that the second 512 words of a virtual code file are page zero. The A and B parameters (which must be specified in hexadecimal) can be used to omit these system pages from the comparison.

An example which compares virtual code files FILE1 with FILE2, omitting the minus pages and displaying up to 30 nonmatching words, is:

COMPARE,FILE1,FILE2,N=30,A=200,B=200.

COPY - COPY FILE

The COPY control statement copies a file to another file. Both physical and virtual files can be copied.

To copy a tape file, or to copy a file to a tape file, the user must assign the tape with a REQUEST statement before the COPY statement. COPY cannot copy multivolume tape files.

Figure 4-6 shows the COPY control statement format. The first two parameters are required and must be in the order shown. All other parameters are optional and can appear in any order.

COPY,inlfn,outlfn,L=len,I=inadr,O=outadr,PACK=packid.	
inlfn	Name of file to be copied.
outlfn	Name of file to contain a copy of all or part of file inlfn. It can be either an existing file or a new file to be created by the utility.
L=len	Hexadecimal number of words to be copied.
I=inadr	Relative hexadecimal word address in file inlfn where copying is to begin, counting the first word of the file as 0. If the I parameter is omitted, inlfn is copied from its beginning.
O=outadr	Relative hexadecimal word address in file outlfn at which copied information is to be placed, counting the first word of the file as 0. This parameter is not valid for tape files. If the O parameter is omitted, the copy begins at the beginning of outlfn.
PACK=packid	Identifier for the pack on which outlfn is to reside. If outlfn already exists on another pack, the system ignores this parameter, copies inlfn to the existing outlfn, and sends a warning message to the controller. If packid is omitted and outlfn does not exist, the system selects a pack and creates outlfn.

Figure 4-6. COPY Control Statement Format

The input file must be attached. If the output file does not exist or is not attached when COPY is called into execution, the utility creates the file. The new file is a local file with the same characteristics as the input file including type, record format, security level, internal characteristics, and length.

When the user specifies a starting location with the O parameter, enough file space must exist following the specified location to contain the copied data. An output file created by COPY has the same length as the input file. If an output file that is longer than the input file must be created, the user must issue a DEFINE or REQUEST statement to create the output file.

The copy operation terminates when reaching the end of the input file or the end of the output file or when the number of words specified by the L parameter has been copied, whichever occurs first. Status and error information from the utility is returned to the dayfile of a batch job or to the terminal of an interactive user. The hexadecimal number of words copied is displayed.

The first 512 words of any virtual file contain the minus page that the system uses to equate virtual addresses with actual mass storage addresses. If the minus page is not to be copied or overwritten, the I and O parameters (which require hexadecimal values) must be used. I=200 and O=200, for example, skip the minus pages. Similarly, the second 512 words of a virtual code file contain page zero for the file; if the zero page is not to be copied, I should be further adjusted to I=400.

DEFINE - CREATE PERMANENT FILE OR MAKE LOCAL FILE PERMANENT

The DEFINE control statement defines a permanent mass storage file. DEFINE can be used to create a permanent file or to make a local file permanent.

DEFINE can ensure that a file is created on a particular pack. It can also determine whether a given pack has adequate space to hold a file of the required size; if not, the utility returns a fatal error code.

Execution of DEFINE, either to create a permanent file or to make a local file permanent, results in appropriate entries in the pack file index. Creation of a permanent file results in allocation of mass storage. The DEFINE statement must not specify a tape file assigned to the job.

DEFINE controls whether mass storage allocated to the file is contiguous at creation and whether the file can be extended.

The NOSEGMENT and NOEXTEND parameters are used to control the continuity of the file. The interaction between the NOSEGMENT and NOEXTEND parameters is as follows:

Extendable File	Segmentable File	Result
No	No	One segment. File cannot be extended.
No	Yes	File created as one or two segments. File cannot be extended.

Extendable File	Segmentable File	Result
Yes	No	File created as one segment. Noncontiguous segments can be added.
Yes	Yes	File created as one or two segments. Noncontiguous segments can be added.

Figure 4-7 shows the DEFINE control statement format. The first parameter must be the file name. File length, if specified, must be the second parameter. All other parameters are optional and can appear in any order. If a local file is made permanent, length and all other parameters are ignored.

Upon successful completion, the message CREATED PERMANENT FILE or EXISTING LOCAL FILE MADE PERMANENT is printed.

DEFINE,ifn/len,ACCESS=acs,TYPE=typ, SECURITY=lvl,PACK=packid,NOEXTEND, NOSEGMENT,RT=rt,MNR=mnr,MXR=mxr,PC=pc, RMK=rmk.		If NOEXTEND is omitted, the file can be extended a percentage of its original length as set by an installation parameter.	
ifn	Name of the new permanent mass storage file created. Ifn must be one through eight letters or digits beginning with a letter (except for the name of a local drop file).	NOSEGMENT	Indicates that the initial file space allocated must be contiguous. If NOSEGMENT is omitted, the system might allocate initial file space in two segments.
/len	Number of small pages allocated for the file (decimal or hexadecimal number between 1 and #FFFF).	RT=rt	Record type (refer to Record Formats in section 2). If RT=rt is omitted, R format is used.
ACCESS=acs	File access permission. If ACCESS=acs is omitted, read and write permission is granted.		F ANSI fixed length. R Record mark delimited. U Undefined. W Control word delimited.
	R Read access. W Write access. RW Read and write or access. WR	MNR=mnr	Minimum record length in bytes. If RT=F is specified, mnr is ignored. If MNR=mnr is omitted, the minimum length is one byte.
TYPE=typ	File type. If TYPE=typ is omitted, the file is a physical data file.	MXR=mxr	Maximum record length in bytes. If RT=F is specified, mnr is the fixed record length. If MXR=mxr is omitted, no maximum record length is set.
	C Virtual code file. P Physical data file.	PC=pc	ASCII padding character used to fill the working storage area. If PC=pc is omitted, blank fill is used.
SECURITY=lvl	Security level (1 through 256) of created file. If SECURITY=lvl is omitted, the security level set by the STORE card or LOGON statement is used.	RMK=rmk	ASCII record delimiting character for R format records. If RMK=rmk is omitted, the installation-defined delimiter is used.
PACK=packid	Identifier of pack on which the file is created. Pack identifiers are six characters long, left-justified, and blank-filled. Excess characters are truncated.		
NOEXTEND	If PACK=packid is omitted, the system selects a pack. Indicates that the file cannot be extended.		

Figure 4-7. DEFINE Control Statement Format

The retention period for the file is an installation option. The SWITCH control statement can be used to specify a particular number of days the file is to be retained on mass storage.

DUMPF - DUMP FILES

The DUMPF control statement dumps public, permanent, and pool files to another pack or to a magnetic tape. At programmer option, the original mass storage file is purged when a file is dumped. Files to be dumped can be selected by user number, pack, residence, file name, or pool name. Selection of files operates by a combination of parameters specified. Further qualification of files to be dumped can be specified by date, time, and type of the last file operation.

A nonprivileged user can only dump attached permanent files or attached pool files. A privileged user can dump all permanent, pool, or public files in the system, with the exception of attached permanent files. Local files are not processed. At the completion of a DUMPF, the dump files created from the original mass storage files become unattached permanent files.

When dumping a file to mass storage, DUMPF maintains a directory file for each user. The directory contains the generated file names of all the dumped files for this user on the mass storage device. The dumped file exists as a private file generated by DUMPF. The contents of this file consists of the data needed for reloading the files in the first block followed by the contents of the file. The format of the directory file and each dumped file are described in figure 4-8.

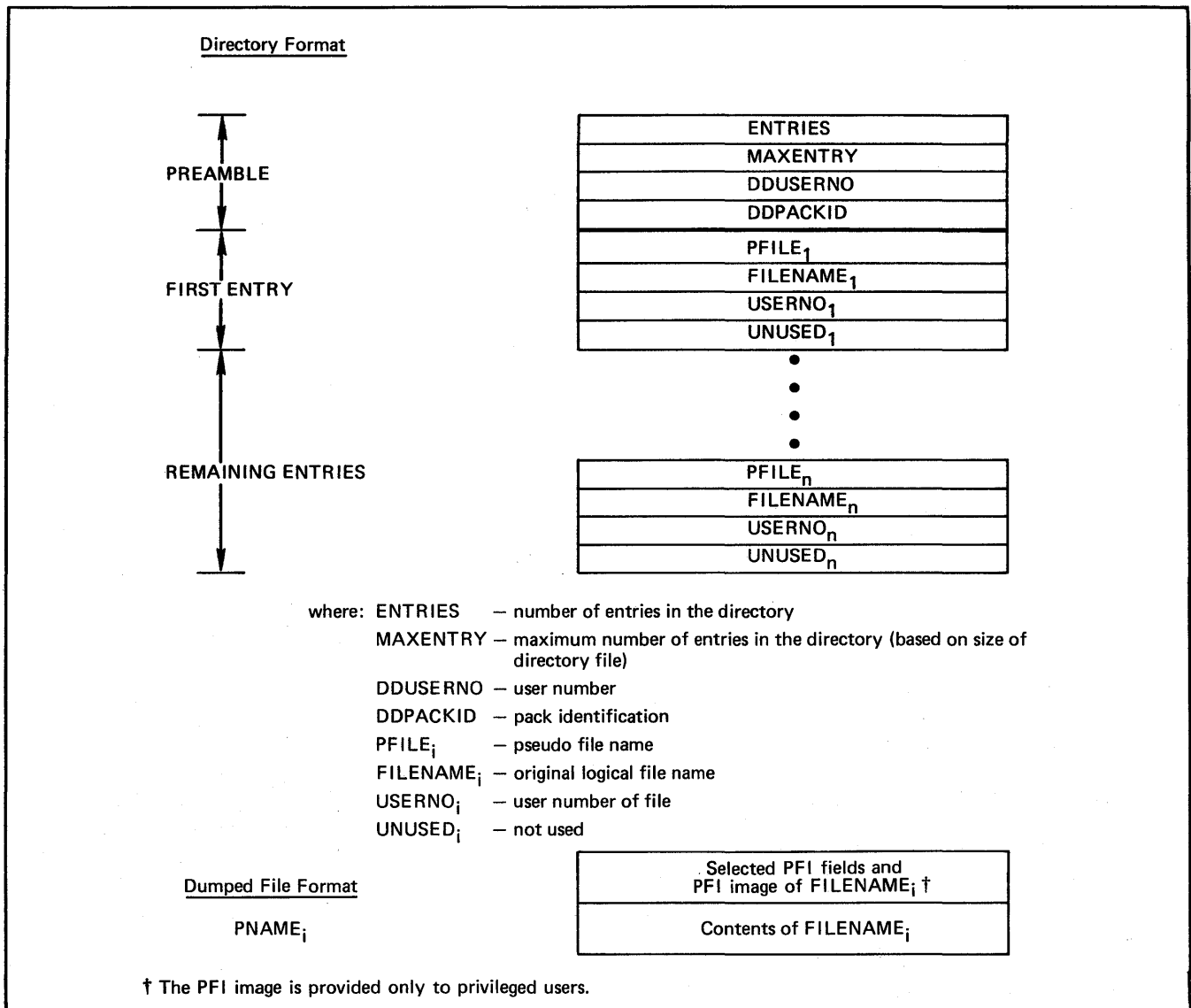


Figure 4-8. Directory/Dumped File Format

TPAKmmnn is referred to as the pseudo file. TPAKmm represents the pack identifier specified by the VSN parameter. nn is the file sequence number which is in hexadecimal.

Currently, a nonprivileged user can dump #FF files, while a privileged user can dump #3FF files. When the sequence number is greater than #FF, the pseudo file is then represented by PAKmmnnn.

When dumping files to mass storage, an entry is created in the directory for each file, lfn. If the file, lfn, already exists in the directory for this user number, the existing lfn is destroyed and the current file, lfn, is created.

DUMPF control statement format is shown in figure 4-9. All parameters are optional and can appear in any order.

<p>DUMPF,DD=device,VSN=id-list,DE=density, RE=days,UN=userno,POOL=pl-list,PN=pkid-list, PF=lfn-list,OP=opts,DT=mmddyy,TM=hhmm, LO=x,OU=lfn/len/dc.</p>		UN=userno	Indicates files to be dumped:
<p>DD=device Dump device:</p>			For a nonprivileged user:
	NT 9-track tape.		userno User number under which DUMPF is executing.
	MT 7-track tape.		For a privileged user:
	MS Pack indicated by VSN parameter.	u-list	List of 1 through 128 user numbers, separated by commas, of private files to be dumped.
	If the DD parameter is omitted, default is an installation option.	ALL	Indicates that all private, pool, and public files are to be dumped.
VSN=id-list	Identification of tape or pack to receive dump:		
	For tapes, a list of 1 through 128 volume serial numbers of tapes. If omitted, the operator assigns a tape.	POOL=pl-list	List of 1 through 128 pools, separated by commas, of pools to be dumped.
	For packs, a list of pack identifiers of one through six characters. Required.	PN=pkid-list	List of 1 through 16 identifiers, separated by commas, of packs to be dumped.
	If sufficient tapes or packs are not specified, the operator is instructed to assign additional devices.		If the PN parameter is omitted, all active packs are searched for files meeting other dump criteria.
DE=density	Density of dump tape:	PF=lfn-list	List of 1 through 128 names, separated by commas, of private or pool files to be dumped.
	LO 200 bpi (7-track tape only).	OP=opts	File characteristics which qualify files selected by UN, PN, POOL, or PF parameters.
	HI 556 bpi (7-track or 9-track).		Options A, C, M, N, X, or P can be specified in any order; commas must not appear between these characters.
	HY 800 bpi (7-track or 9-track).		A Files accessed since date and time specified.
	PE 1600 bpi (9-track only).		
	If the DE parameter is omitted, default is an installation option.		
RE=days	Tape retention period (1 through 999 days).		
	If the RE parameter is omitted, default is an installation option.		

Figure 4-9. DUMPF Control Statement Format (Sheet 1 of 2)

C	Files created since date and time specified.	TM=hhmm	Time to modify the A, C, or M option, in a format indicating hours and minutes in a 24-hour clock.
M	Files modified since date and time specified.		
N	Reverse the meaning of any A, C, or M specified. That is, the appearance of N changes the meaning of A from accessed to not accessed, the meaning of C from created to not created, and the meaning of M from modified to not modified.	LO=x	Indicator of type of audit information to be written. F Full information. P Partial information. Default.
X	Files expired. That is, files whose creation date plus retention period specifies a date preceding the current date.	OU=lfm/len/dc	File to which dump status information is to be written. lfm Name of file. Must be one through eight letters or digits beginning with a letter. Default is OUTPUT. len Number of small pages in file. When /len is omitted, default is #40. dc Disposition code indicating processing of file: PR Print on any available printer at the end of the utility. When /dc is omitted, the utility does not cause the file to be printed.
P	Indicator that file is to be purged from mass storage after it is dumped successfully.		
If the OP parameter is omitted, default is none of these options.			
DT=mmddyy	Date to modify the A, C, or M option, in format indicating month of year, day of month, and the last two digits of the year.		
If the DT parameter is omitted, default is the current date.			

Figure 4-9. DUMPF Control Statement Format (Sheet 2 of 2)

Parameters work in logical combinations to determine files to be dumped. When PN, UN, PF, and POOL all are omitted, only private files associated with the user are dumped. When more than one of these parameters is specified, a file must meet all criteria specified before it is dumped. Similarly, the options specified by the OP parameter operate in combination to select files. A parameter OP=CM, for example, selects files created or modified after a specified date and time; a parameter OP=NCM selects files that have not been created and have not been modified after a specified date and time. The UN and POOL parameters interact as shown in table 4-2.

A dump tape or file produced by a nonprivileged DUMPF contains the contents of selected PFI fields prior to opening each file being dumped; therefore, the access fields are not updated on the dump tape/file but are updated on the file index maintained by the system. An attempt to reload the files by date and time of last access uses the original values for those fields for the comparison.

A dump tape or file produced by a privileged DUMPF contains the contents of selected PFI fields and a copy of the unformatted PFI after opening each file being dumped; therefore, the access fields are updated on both the dump tape/file and the file index maintained by the system. An attempt to reload the files by date and time of last access uses the updated values for those fields for the comparison.

Pool files can be dumped by any user authorized to access the pool when the POOL parameter is specified. Only the pool boss can execute with OP=P to purge the mass storage copy of the file. A PATTACH control statement must precede use of the POOL parameter for a nonprivileged user dump of pool files.

DUMPF can execute concurrently with other tasks, including other DUMPF tasks. If a file cannot be dumped, the utility writes an appropriate message for the file specified by the OU parameter and continues with the dump of other files selected. Attached or unattached status is not reported on the output file.

Dumping files to tape produces an ANSI-labeled tape. The DUMPF utility displays appropriate messages for operator action.

EDITPUB - ADD OR DESTROY PUBLIC FILE

The EDITPUB control statement is valid only for privileged user numbers. It adds or destroys a file from the ownership category of public. Files processed with EDITPUB must be attached private or pool files; they cannot be tape files. Only the pool boss can issue an EDITPUB statement for a pool file.

EDITPUB format is shown in figure 4-10.

When the utility is called with the L parameter from an interactive terminal, it displays the name of each public file in turn and waits for one of the following terminal user responses:

D	Destroy file.
carriage return	Retain file.
STOP	Terminate utility.

$\text{EDITPUB, } \left\{ \begin{array}{c} D=\text{ifn-list} \\ L \\ \text{VRI=index.} \end{array} \right\}, N=\text{ifn-list}, P=\text{ifn-list},$	
L	Files to be destroyed are specified interactively (interactive call only).
D=ifn-list	List of public files to be destroyed (1 through 16 names, separated by commas).
N=ifn-list	List of files to be added to the public file list without privileged status (1 through 16 names, separated by commas).
P=ifn-list	List of files to be added to the public file list with privileged status (1 through 16 names, separated by commas).
VRI=index	Index into the Variable Rate Table in the range 1 through 255, for public files being added with the call. If VRI=index is omitted, the system uses an index of 0.

Figure 4-10. EDITPUB Control Statement Format

Files cannot be destroyed while they are open to any task. If the system cannot destroy a file, it returns error messages to the dayfile of a batch job or to the terminal of an interactive user.

If a file being made public has the same name as an existing public file (that is, if a file is being replaced), the destroy and add operations can be performed through a single call. For example, to replace public files X and Y, the following control statement is appropriate:

EDITPUB(D=X,Y,N=X,Y)

If the VRI parameter is used, at least one of the N or P parameters must be used. In this case, all files being made public in this control statement must be controllees, and the VRI parameter will apply to all. Files made public using the VRI parameter will not retain read or write access.

If both the L and VRI parameters are used from an interactive terminal, and the user response indicates that any file is to be retained, the VRI for that file is not reset to the VRI parameter value. The VRI file index entry for any file is 0 until modified by a VRI specification in EDITPUB.

EXIT - SET ABNORMAL TERMINATION PATH

The EXIT control statement is valid only in a batch job. It is executed directly by the batch processor. It establishes a control path to be followed in the event of abnormal job termination.

Whenever a threshold value test fails (see TV control statement), the batch processor searches subsequent statements and resumes execution at the control statement following the first EXIT encountered. If no EXIT control statement exists, the job ends abnormally. If the EXIT statement is encountered during normal job advancement to the next control statement, the job ends normally.

EXIT control statement format is shown in figure 4-11. More than one EXIT control statement can appear in a job.

EXIT.

Figure 4-11. EXIT Control Statement Format

The threshold value is set to 255 when an EXIT control statement establishes the execution path.

If control transfers to the path established by an EXIT control statement because the job time limit is reached, a short amount of time is made available to the job for use by the batch processor. In this case, user job tasks cannot be performed after the EXIT statement.

FILES - LIST FILES

The FILES control statement lists status information about specified public files, private files, and pool files. Private files include local files at the suffix only and all attached or unattached permanent files.

The FILES control statement format is as shown in figure 4-12. All parameters are optional, and, with the exception of a list of private files, can appear in any order. Omission of all parameters is equivalent to FILES (PRIVATE=*).

All specified files are listed in alphabetical order. Appropriate headings identify different types of information:

Heading	Meaning
NAME	File name.
D	Duplicate file name flag. An asterisk in this column indicates that at least one other file exists with the same name and owner.
OWNER	File ownership for mass storage files: public (*PUBLIC), permanent (*PERM), local (*LOCAL), or pool (poolname). For tape files, this field contains the volume serial number.

Heading	Meaning
SUF	The suffix (if any) to which the file is attached (local and permanent files only).
ACS	Access permission: read (R), write (W) or read and write (RW).
TYPE	File type: virtual code (VC) or physical data (PD).
DT	Device type: mass storage (MS), seven-track tape (MT), or nine-track tape (NT).
FC	File category: system-generated drop file (S), batch file (B), user file (U), and not defined (N).
BT	Blocking type: character count (C) or non-SIL file (blank).
RT	Record type: ANSI fixed length (F), record mark delimited (R), undefined (U), or control word delimited (W).
ORI.DATE	Origin date (the date the file was created).
UNIT	Logical unit number of the device on which the file resides.
LEN	Actual length of each segment of the file (decimal number of 512-word blocks).
SADDR	Hexadecimal physical sector address of the segment. This column is not displayed at a terminal.
RP	Retention period (the number of days the file is to be retained). This column is not displayed at a terminal.

FILES, Ifn-list, PUBLIC= { Ifn-list }, PRIVATE= { Ifn-list }, POOL=poolname, Ifn-list, L=Ifn.	
Ifn-list	List of files to have status information written.
PUBLIC= { Ifn-list }	Public files to be listed:
{ Ifn-list }	List of 1 through 255 public file names, separated by commas.
{ *	All public files.
PRIVATE= { Ifn-list }	Private files to be listed:
{ Ifn-list }	List of 1 through 255 private file names, separated by commas.
{ *	All permanent files for the submitting user and local files for the issuing task suffix. This is the default.

Figure 4-12. FILES Control Statement Format (Sheet 1 of 2)

POOL=poolname,lfn-list

Pool files to be listed:

poolname Name of pool to which user is attached.

lfn-list List of 1 through 255 file names in pool poolname. If omitted, all files in pool are listed.

The POOL parameter can be repeated within the parameter list.

L=lfn

Name of file to which output is to be written. Must be one through eight letters or digits beginning with a letter. Any existing file with the same name is destroyed.

Default in batch mode is OUTPUT. Default in interactive mode returns output to the terminal.

Figure 4-12. FILES Control Statement Format (Sheet 2 of 2)

The same file name can be specified more than once and might appear in the output more than once, since file names need be unique only in relation to all other files with the same ownership. On output, files with duplicated names are listed in the order: local, permanent, pool, and public.

From an interactive terminal, this utility can be called by name alone and the user is prompted for parameters. Information is displayed 15 lines at a time. When output exceeds display size, the programmer is required to enter CONTINUE to continue the display or enter END to terminate the display.

Figure 4-13 shows an example of FILES output from a terminal.

GIVE - CHANGE FILE OWNER

The GIVE control statement changes the owner of an attached private or pool file. It cannot change the owner of a tape file. Any file referenced must not be open at the time this utility executes. Only the pool boss can issue a GIVE statement for a pool file.

The current file owner can give one or more files to:

- Another user number. In this instance the file ownership category remains private, but the user number changes. The file becomes an unattached permanent file belonging to the new owner.
- A pool. In this instance the file ownership category changes to pool. No user, including the pool boss, can then access the file without attaching to the pool through the PATTACH utility.
- The public file list. The file is given to user number 000000. Only a privileged user can create a public file.

Once GIVE references a file, it is no longer accessible through the old user number.

Figure 4-14 shows the GIVE format. Parameters can appear in any order.

Public files cannot be referenced by GIVE. Further, any file name referenced by GIVE cannot have the same name as a public file, unless it is being given to a pool.

NAME	D	OWNER	SUF	ACS	TYPE	DT	FC	BT	RT	ORLDATE	LEN	UNIT
42FORTRA		*PERM	A	R	VC	MS	U	C	U	04/14/80	00140	01
FORTTRAN		*PUBLIC		R	VC	MS	U	C	U	04/01/80	00040	01
OUTPUT		*LOCAL	A	RW	PD	MS	U	C	R	04/14/80	00001	05
TAPE1		102342	A	R	PD	MT	U	C	F	04/14/80		03

Figure 4-13. FILES Sample Output

GIVE, { * lfn-list }, { U=newown POOL=poolname,SHARE=perm }	
*	Indicator that all attached private files associated with the user are to change ownership.
lfn-list	List of 1 through 16 files names, separated by commas, of files whose ownership is to change. Must not include files with the same name as a public file.
U=newown	User number of new owner of private files. If newown is 000000 and the user is privileged, the file becomes public.
POOL=poolname	Name of existing pool to receive files listed.
SHARE=perm	Access allowed for files being given to a pool:
R	Any user attaching to pool can read or execute the file. Default.
W	Any user attaching to pool can write or execute the file.
RW	Any user attaching to pool can read, write, or execute the file.
NONE	No access possible.

Figure 4-14. GIVE Control Statement Format

LOAD - CREATE CONTROLLEE FILE

The LOAD control statement transforms object modules into a virtual code controllee file suitable for execution.

Figure 4-15 shows the LOAD control statement format. All parameters are optional, although subparameters cannot be separated. Any list of files to be loaded must appear first; otherwise, parameters can appear in any order. Any number of items can appear in the lists associated with the LIBRARY, EQUATE, DEBUG, and the GRxx parameters. Multiple GRxx parameters can appear.

If using the loader interactively from a terminal, the user must type SPACE and NEWLINE to indicate no options or to terminate options. Figure 3-7 is an example of interactive LOAD utility usage.

Input files to the loader must contain object modules. An input file can be either an object code file produced by a CYBER 200 assembler or compiler or a modmerge file produced by OLE. All input files must be local or attached permanent or pool files. The file specified as the output can be a local file or an attached permanent file.

The number of optional loader files cannot exceed 13. Optional loader files are SYSLIB, library files, and user files. The number of user files cannot exceed 10. The size of the controllee file is reduced at job termination.

LOAD, lfn₁, ..., lfn_n, CTROLEE= lfn/len,
 CDF=dlen, OUTPUT=lfn/len, LIBRARY=lib₁, ..., lib_n,
 EQUATE=sub₁, nam₁, ..., sub_n, nam_n,
 ENTRY=ept, DEBUG=mod₁, ..., mod_n,
 VR=string, ORIGIN=bitadr,
 GRSP=mod₁, ..., mod_n, bitadr,
 GRLP=mod₁, ..., mod_n, bitadr,
 GROS=com₁, ..., com_n, bitadr,
 GROL=com₁, ..., com_n, bitadr,
 GRLPALL=Δ, DSA=bitadr, LO=X.

EQUATE=sub_i, nam_i

List of external reference pairs. The second name in a pair replaces the first name during linking.

Common names must be preceded by an asterisk. An asterisk alone indicates blank common.

ENTRY=ept

Name of an entry point in a loaded module at which execution is to begin (the transfer address). If ENTRY=ept is omitted, MAIN. ΔΔΔ is used.

DEBUG=mod_i

List of modules for which the debug version is to be loaded.

ORIGIN=bitadr

Virtual bit address at which loading is to begin. The loader adjusts this address upward to a page boundary if necessary. If ORIGIN=bitadr is omitted, loading begins at address #8000.

GRSP=mod_i, bitadr

List of blocks to be loaded as group at beginning of 512-word block (small page) boundary.

mod_i can be a list of modules or a list of common blocks. An asterisk must prefix common block names. An asterisk alone denotes blank common. Modules and common blocks must be grouped by separate parameters.

bitadr is the bit address of the small page. If bitadr is omitted, the next available small page is used.

GRLP=mod_i, bitadr

List of modules or common blocks to be loaded as group at beginning of large page (128-block) boundary.

mod_i can be a list of modules or a list of common blocks. An asterisk must prefix common block names. An asterisk alone denotes blank common. Modules and common blocks must be grouped by separate parameters.

bitadr is the bit address of the large page. If bitadr is omitted, the next available large page is used.

lfn_i List of input files (1 through 10 file names, separated by commas). An input file is an object code file produced by a compiler or assembler or a modmerge file produced by OLE. If no input files are listed, LOAD uses file BINARY.

CTROLEE=lfn/len lfn is the file to which LOAD writes the executable virtual code. If the parameter is omitted, LOAD writes the code on file GO.

len is the number of 512-word blocks allocated for the file. If len is omitted, #102 blocks are allocated. Unused file space is released at job termination.

CDF=dlen Number of 512-word blocks in the drop file created when this controllee file is executed. (This value is stored in word #20 of the minus page.) If CDF=dlen is omitted, word #20 of the minus page is zero and the system determines the drop file size.

OUTPUT=lfn/len lfn is the local or attached permanent file to which LOAD writes the load map. If this parameter is omitted, LOAD writes the map on the local file OUTPUT.

len is the number of 512-word blocks allocated for the file. If len is omitted, #25 blocks are allocated. Unused file space is released at the end of map construction.

LIBRARY=lib_i List of library files from which LOAD satisfies external references. (OLE creates library files.) If LIBRARY=lib_i is omitted, LOAD searches only file SYSLIB.

Figure 4-15. LOAD Control Statement Format (Sheet 1 of 2)

GROS=com _i ,bitadr	List of common blocks to be grouped and relocated on a small page boundary, but for which the controllee file has no reserved space.	GRLPALL= Δ	Indicates that all code, data base, and labeled common is to be grouped on large page boundaries. A blank must follow the = in the parameter.
	com _i can be a list of named common blocks or an asterisk indicating blank common. Named and blank common must be grouped by separate parameters.	VR=string	String of one through eight ASCII characters to be stored left-justified and blank filled in register #A. The string cannot contain the characters , .) and blank.
	bitadr is the bit address of the small page. If bitadr is omitted, the common blocks are loaded after previously loaded items.	DSA=bitadr	Virtual address at which the dynamic stack begins. It must be a page boundary. If DSA=bitadr is omitted, the dynamic stack begins at the last virtual address allocated.
GROL=com _i ,bitadr	List of common blocks to be grouped and relocated on a large page boundary, but for which the controllee file has no reserved space.	LO=X	Indicates that the load map should include a common block and entry point cross-reference list. All common blocks and entry points are listed alphabetically with the modules that reference them. If LO=X is omitted, the common block and entry point cross-reference is omitted.
	com _i can be a list of name common blocks or an asterisk indicating blank common. Named and blank common must be grouped by separate parameters.		
	bitadr is the bit address of the large page. If bitadr is omitted, the common blocks are loaded after previously loaded items.		

Figure 4-15. LOAD Control Statement Format (Sheet 2 of 2)

Figure 4-16 shows the format of a virtual code file.

The first page of a controllee file is its minus page. The loader initializes the information stored in the minus page.

Labeled common is preset to all zeros and data is loaded as specified. Common blocks that are multiples of 512 words are forced to small page boundaries. Numbered and blank common areas are mapped into the drop file map. On the first access to the common area, the user obtains a page created by the operating system and initialized to #000C1F1C.

The loading process loads modules from files in the order they are listed on the LOAD control statement. Unsatisfied externals referenced by those modules are

satisfied from the libraries specified by the LIBRARY parameter. If no library is specified, or if unsatisfied externals remain after a search of all specified libraries, the file SYSLIB is searched for module names that would satisfy externals.

The user can direct the loader to link external references to routines on a library containing user-written debugging versions of specified routines. Use of the DEBUG parameter changes the linkage, so that a reference to NAME, for example, is linked to debugging routine NAMEQ. Any reference to NAME within NAMEQ is linked to NAME, although other references to NAME are linked to NAMEQ. Both versions of a routine must exist on the same library.

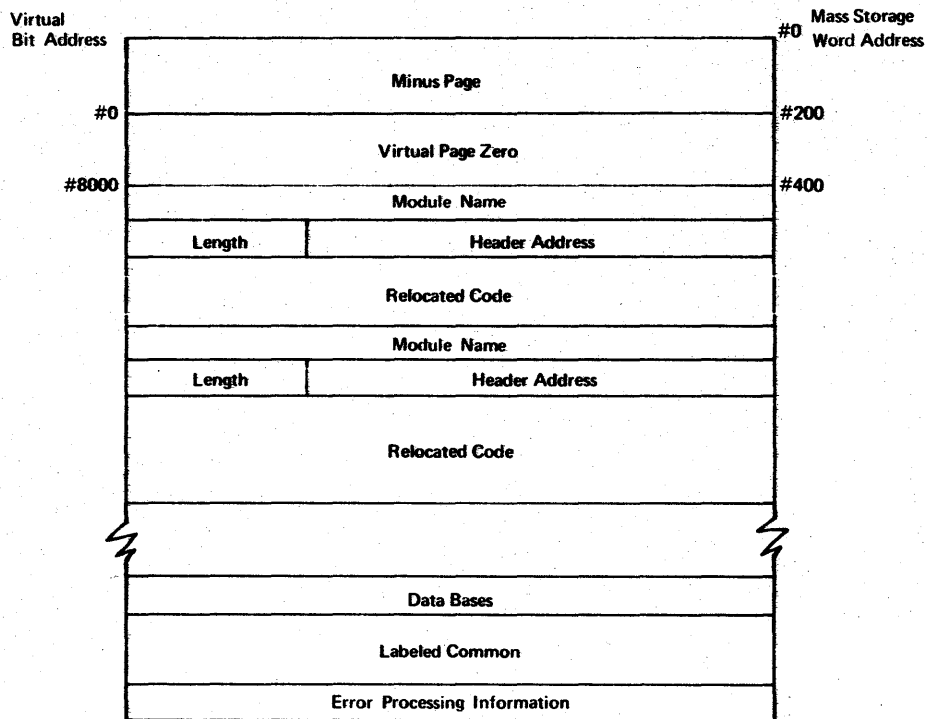


Figure 4-16. Virtual Code File Format

The EQUATE parameter can also change linkages. It allows substitution of external references so that the user can change external entry point names or common area names at load time.

Loading begins at the virtual address specified by the ORIGIN parameter, which is adjusted upward if necessary to a page boundary.

The GRSP or GRLP parameters designate modules or common blocks to be grouped and loaded in small or large pages. Code modules and common blocks initialized with data can be grouped by the GRSP and GRLP parameters, but not by the GROS or GROL parameters.

The GROS and GROL parameters designate groups of modules or common blocks whose origin is to be on a small or large page boundary. The group is not mapped into the controllee file nor the drop file, but a virtual address range is reserved for the group. This virtual range can be mapped in during execution.

Note that although these options provide a grouping capability, no ordering is implied.

The user must specify separate grouping parameters for modules, for named common blocks, and for blank common.

The user specifying a bit address on a GRxx parameter must ensure that that address has not been allocated.

The loader produces a load map showing the locations of routines and data in the program. The map is written to the file specified in the OUTPUT parameter with ASCII carriage control characters suitable for printing.

With the VR parameter, the user can provide an identification recognizable in a dump. The character string specified by this parameter is stored in register #A of the zero page. The loader also stores the date and time of controllee file creation in registers #B and #C. The date format is the eight characters mm/dd/yy representing the month, day of month, and last two digits of the year. The time format is the eight characters hh.mm.ss representing the hour on a 24-hour clock, minutes of the hour, and seconds.

The dynamic stack address referenced by the DSA parameter is concerned with temporary working space available in the register file. One typical use of the dynamic stack is for saving registers over a call. They are sometimes referred to as a data base. The dynamic stack address is defined as the current stack pointer, starting at the DSA address; the dynamic stack pointer is #180 bits (six words) after the current stack pointer. If the current stack pointer is at #200000, the dynamic stack pointer is at #200180. The loader assigns a default DSA following the last virtual address allocated. The dynamic stack address is always printed on load maps.

LOADPF - RELOAD FILES

The LOADPF control statement reloads mass storage files that have been dumped to a tape or to a pack. Files to be loaded can be selected by permanent file name or by pool file name. Further qualification of files to be reloaded can be specified by date, by time, or by last access.

A nonprivileged user can load only files owned by the user. Pseudo files (refer to the DUMPF control statement description for a description of pseudo files) are attached and then returned by LOADPF. Files loaded from mass storage become unattached permanent files. Local files are not processed. A privileged user can load all files in the system. Pool files can be reloaded by any user, but only the pool boss or privileged user can restore file status as a member of a particular pool. If the system cannot then

restore pool status because the pool no longer exists or another file exists with that name, the reloaded file remains a permanent file.

Figure 4-17 shows the LOADPF control statement format. The dump device must be adequately specified by DE and DD. All other parameters are optional and can appear in any order.

LOADPF,DD=device,VSN=id-list,DE=density,UN=userno,POOL=pl-list,PF=fn-list,OP=opts,DT=mmddyy,TM=hhmm,PN=pkid-list,LO=x,OU=fn/len/dc.

DD=device	Device on which dump files exist:
NT	9-track tape
MT	7-track tape
MS	Pack indicated by VSN parameter
	If the DD parameter is omitted, default is an installation option.
VSN=id-list	List of 1 through 128 pack identifiers separated by commas, on which dump files exist; or of volume serial numbers of tapes. Required parameter.
DE=density	Density of dump tape:
LO	200 bpi
HI	556 bpi
HY	800 bpi
PE	1600 bpi
UN=userno	Indicator of files to be reloaded:
	For a nonprivileged user:
userno	User number under which LOADPF is executing.
	For a privileged user:
u-list	List of 1 through 128 user numbers, separated by commas, of files to be reloaded.
ALL	Indicates that all files are to be reloaded.
POOL=pl-list	List of 1 through 128 pool names, separated by commas, of pools whose files are to be reloaded.
PF=fn-list	List of 1 through 128 file names, separated by commas, of files to be reloaded.
OP=opts	File characteristics which qualify files selected by UN, PN, POOL, or PF parameters.
	Options A, C, M, N, X, and R can be specified in any order; commas must not appear between these characters.
A	Files accessed since date and time specified.
C	Files created since date and time specified.
M	Files modified since date and time specified.
N	Reverse the meaning of any A, C, or M specified. That is, the appearance of N changes the meaning of A from accessed to not accessed, the meaning of C from created to not created, and the meaning of M from modified to not modified.

Figure 4-17. LOADPF Control Statement Format (Sheet 1 of 2)

	X	Files expired. That is, files whose creation date plus retention period specifies a date preceding the current date.
	R	Indicator that existing file with the same name as a file being reloaded is to be destroyed and the dumped file is to take its place.
		If omitted, a file with a duplicated name is not to be reloaded.
		If the OP parameter is omitted, default is none of these options.
DT=mmddyy		Date to modify the A, C, or M option, in format indicating month of year, day of month, and the last two digits of the year.
		If the DT parameter is omitted, default is the current date.
TM=hhmm		Time to modify the A, C, or M option, in a format indicating hours and minutes on a 24-hour clock.
		If the TM parameter is omitted, default is 0000 which is midnight preceding the day specified by the DT parameter.
PN=pkid-list		List of 1 through 16 pack identifiers, separated by commas, of packs on which files are to be reloaded.
		If the PN parameter is omitted, default is all active packs.
LO=x		Indicates type of audit information to be written:
	F	Full information.
	P	Partial information. Default.
OU=lfm/len/dc		File to which reload status information is to be written.
	lfm	Name of file. Must be 1 through 8 letters or digits beginning with a letter. Default is OUTPUT.
	len	Number of small pages in file. When /len is omitted, default is #40.
	dc	Disposition code indicating processing of file:
	PR	Print on any available printer at the end of the utility.
		When /dc is omitted, the utility does not print the file.

Figure 4-17. LOADPF Control Statement Format (Sheet 2 of 2)

Parameters work in logical combinations to determine files to be loaded. When PN, UN, PF, and POOL are all omitted, only permanent files associated with the user are loaded. When more than one of these parameters is specified, a file must meet all criteria specified before being loaded. Similarly, the options specified by the OP parameter operate in combination to select files. A parameter OP=CM, for example, selects files created or modified. The UN and POOL parameters interact as shown in table 4-2.

A dump tape or file produced by a nonprivileged DUMPF contains the contents of selected PFI fields prior to opening each file being dumped; therefore, the access fields are not updated on the dump tape/file but are updated on the file index maintained by the system. An attempt to reload the files by date and time of last access uses the original values for those fields for the comparison.

A dump tape or file produced by a privileged DUMPF contains the contents of selected PFI fields and a copy of the unformatted PFI after opening each file being dumped; therefore, the access fields are updated on both the dump tape/file and the file index maintained by the system. An attempt to reload the files by date and time of last access uses the updated values for those fields for the comparison.

LOADPF can execute concurrently with other tasks, including other LOADPF tasks. It produces, as output, a list file containing the names of files loaded, and any appropriate error messages, including the file name of the file being processed. Attached or unattached status is not reported on the output file.

NORERUN - SET NORERUN STATUS

The NORERUN control statement is valid only within a batch deck. It is executed directly by the batch processor. It changes the default status for the file created by the STORE card from rerun to norerun, such that when the system is brought up after a system failure, the batch input file is destroyed.

Figure 4-18 shows the NORERUN format. An example of NORERUN use is shown in figure 4-19.

NORERUN.

Figure 4-18. NORERUN Control Statement Format

OLE - OBJECT LIBRARY EDITOR

The OLE control statement operates with modules produced by assembly or compilation of source programs. It can be used to create either of the following:

- A library file. The library contains a directory of module names and entry points, as well as the modules, and can be used by the loader to satisfy externals. When referencing the library by the LIBRARY parameter LOAD, the loader can selectively load modules from the library to satisfy externals or the ENTRY parameter of LOAD.
- A modmerge file. This file is a collection of modules without a directory. When referencing the file in a LOAD control statement, all modules are loaded, subject to the loading order specified in the LOAD control statement. A modmerge file offers the programmer convenience in referencing a group of modules, and also overcomes the restriction of a limit of 10 files that can be referenced in a LOAD control statement.

OLE can also be used to list modules and characteristics of the modules on library or modmerge files independent of any file creation.

OLE produces a single file in either library or modmerge file format. As many as 50 input files can be specified. Input files must contain a library, one or more modules produced by a CYBER 200 assembler or compiler, or a modmerge format file of several modules. Input files must be attached, and output files are local. The total number of modules and entry points cannot exceed 1500.

Figure 4-20 shows the OLE format. Parameters can appear in any order, but subparameters cannot be separated.

During OLE execution, modules are placed in the new file in the order they are encountered in the input files as listed with the INPUT parameter. Any modules specified by the OMIT parameter are not made a part of the new file. If duplicate module names exist in the input files, only the first module encountered is written to the new file.

POOL FILE UTILITIES

The seven pool file utilities create, access, and destroy a pool of files. Section 2 explains how a pool is created and used.

All of these utilities send error and status messages either to the dayfile of a batch job or to the terminal of an interactive user.

Pool utilities follow in alphabetical order.

```

STORE...
RESOURCE,TL=8.
.
.
COMMENT. JOB RERUNS IF A FAILURE OCCURS HERE
NORERUN.
.
.
COMMENT. JOB DOES NOT RERUN IF A FAILURE OCCURS HERE
RERUN.
.
.
COMMENT. JOB RERUNS IF A FAILURE OCCURS HERE
.
.
6/7/8/9 card

```

Figure 4-19. NORERUN/RERUN Example

OLE,INPUT=lfm-list, { NEWLIB=liblfn MODMERGE=modlfn }, OMIT=sfn,mod-list,LIST=opt,OUTPUT=lfm/len.	
<u>INPUT</u> =lfm-list	List of 1 through 50 file names, separated by commas, whose modules are to be written to the file specified by the NEWLIB or MODMERGE parameter. If the INPUT parameter is omitted, only the parameters LIST and OUTPUT are valid.
<u>NEWLIB</u> =liblfn	Name of file to contain new library being created. Must be 1 through 8 letters or digits beginning with a letter and can duplicate a file name specified with the INPUT parameter. If the NEWLIB and MODMERGE parameters are omitted and INPUT is specified, default is NEWLIB.
<u>MODMERGE</u> =modlfn	Name of file to contain modmerge file being created. Must be one through eight letters or digits beginning with a letter. Can duplicate the name of a file specified with the INPUT parameter. No default name exists.
<u>OMIT</u> =sfn,mod-list	Indicator that listed modules (mod-list) of file sfn are to be omitted from the library or modmerge file. One OMIT parameter can be specified for each input file.
<u>LIST</u> =opt	Indicator of files to be listed by module names, length, creation date, and entry point names: lfm-list List of file names, separated by commas, whose contents are to be listed. The destination file is listed without being specified. 0 Suppress all listings. If the LIST parameter is omitted, only the library or modmerge file is listed. Output appears on the file specified by the OUTPUT parameter.
<u>OUTPUT</u> =lfm/len	File to which listing is to be written. lfm Name of file. Must be one through eight letters or digits beginning with a letter. When the OUTPUT parameter is omitted, default is OUTPUT. len Number of small pages in file. When /len is omitted, default is #10.

Figure 4-20. OLE Control Statement Format

PACCESS POOL UTILITY

Only the pool boss can execute the PACCESS utility. It establishes the list of users authorized to access all files in the pool. The format is shown in figure 4-21.

PACCESS,poolname,USER= $\left\{ \begin{array}{c} \text{u-list} \\ * \end{array} \right\} .$	
poolname	Name of the existing pool.
USER= $\left\{ \begin{array}{c} \text{u-list} \\ * \end{array} \right\}$	Identification of users who are to have pool access.
*	Universal access to all users.
u-list	List of 1 through 32 user numbers, separated by commas, who can access the pool.

Figure 4-21. PACCESS Control Statement Format

PATTACH POOL UTILITY

The PATTACH utility attaches an authorized user to a pool. It is required before any file in the pool can be accessed. The format is shown in figure 4-22.

PATTACH,poolname.	
poolname	Name of existing pool to which user is to be attached.

Figure 4-22. PATTACH Control Statement Format

One user can be attached to as many as four pools simultaneously. Depending on the parameters used by the pool boss when a file was given to the pool, access to any given pool file might be limited to reading only.

PCREATE POOL UTILITY

The PCREATE utility establishes a pool. The user defining a pool name becomes the pool boss and has responsibility for entering files in the pool, specifying who can use the pool, and destroying the pool. The format is shown in figure 4-23.

PCREATE,poolname.	
poolname	Name of pool to be established. Must be 1 through 8 letters or digits beginning with a letter.

Figure 4-23. PCREATE Control Statement Format

PDELETE POOL UTILITY

The PDELETE utility removes users from the list of authorized users. Only the pool boss can execute it. The format is shown in figure 4-24.

PDELETE, poolname, USER= $\left\{ \begin{array}{c} \text{u-list} \\ * \end{array} \right\} .$	
poolname	Name of existing pool.
USER= $\left\{ \begin{array}{c} \text{u-list} \\ * \end{array} \right\}$	Identification of users whose access permission is to be rescinded:
*	Universal access removed.
u-list	List of 1 through 16 user numbers, separated by commas, to be removed. Invalid when universal access is set.

Figure 4-24. PDELETE Control Statement Format

A user cannot be removed from the list of authorized users while attached to the pool. Individual users cannot have their access to a pool deleted if universal access has been granted.

PDESTROY POOL UTILITY

The PDESTROY utility destroys a pool. It can only be executed by the pool boss, who has the responsibility of first removing all files from the pool. The format is shown in figure 4-25.

PDESTROY,poolname.	
poolname	Name of existing pool to be destroyed

Figure 4-25. PDESTROY Control Statement Format

The pool cannot be destroyed while users or the pool boss are attached or while any files exist in the pool. Any pool file to be preserved as a permanent file must be copied to a private file before being removed from the pool.

PDETACH POOL UTILITY

The PDETACH utility detaches a user from a pool. It should be executed when files in a pool are no longer required. The format is shown in figure 4-26.

PDETACH, poolname.

poolname Name of pool from which user
 is to be detached.

Figure 4-26. PDETACH Control Statement Format

PFILES POOL UTILITY

The PFILES utility produces a list of information about pools in the system. Specific information listed depends on the parameters selected. The format is shown in figure 4-27.

PFILES, { poolname
 USER=userno }.

poolname Name of existing pool for which
 authorized users are to be listed.

USER= { u-list
 * } Identification of pool bosses:

 * All pool names are to
 be listed along with
 the pool boss and
 number of users
 currently attached to
 each pool.

 u-list List of 1 through 16
 pool boss user
 numbers, separated by
 commas. All pools
 belonging to each pool
 boss are listed by
 name with the number
 of users currently
 attached to the pool.

Figure 4-27. PFILES Control Statement Format

If information being reported exceeds the size of the display screen of an interactive terminal, the last line of the display instructs the terminal user to enter MORE or YES to continue the display.

Figure 4-28 shows output examples.

A. PFILES(USER=*)

POOL NAME	POOL BOSS	USER COUNT
DEMO	300299	0
TESTPOOL	333322	2
STESTS	333333	0

PFILES UTILITY COMPLETED

B. PFILES(DEMO)

USERS GRANTED ACCESS TO POOL

300045	300047	300034	300089
000022			

Figure 4-28. PFILES Sample Output

PURGE - DESTROY PERMANENT OR POOL FILES

The PURGE control statement releases mass storage space assigned to one or more permanent files. It can also be used by a pool boss to destroy pool files. The utility deletes appropriate entries from the pack file index for packs on which files reside, and releases space for reassignment. The PURGE control statement format is shown in figure 4-29.

PURGE, lfn-list, CL=POOL, ST=xxx, optional
CYBER front-end parameters.

lfn-list List of 1 through 16 CYBER 200
 permanent file names, separated by
 commas, to be purged, or the name
 of one CYBER front-end file to be
 purged. CYBER 200 file names
 must be 1 through 8 letters or digits,
 beginning with a letter (except for
 drop files). lfn-list must appear
 before other parameters. For pool
 files, list of 1 through 16
 CYBER 200 file names, separated by
 commas, of files named in attached
 pools. The CL=POOL parameter is
 also required. Files can be members
 of different pools.

Private and pool files cannot be
mixed in a single list.

CL=POOL Optional parameter that indicates
 that all files named in lfn-list are
 CYBER 200 pool files. The user
 must be the pool boss and be
 attached to all pertinent pools.

ST=xxx Optional parameter that indicates
 that the file in lfn-list is a CYBER
 front-end file resident on the
 mainframe designated by the ST
 parameter.

(The ST parameter is not used when
CYBER 200 files are to be purged.)

optional Refer to the Link Station or
CYBER Access Station Reference Manual.
front-end
parameters

Figure 4-29. PURGE Control Statement Format

If the ST parameter is specified, PURGE is effective for a CYBER file on the designated mainframe.

If a permanent file lfn, specified by PURGE, is currently attached, the file is demoted from permanent to local and is governed by all rules pertaining to local files.

For a successful PURGE of pool files, the user must be the pool boss for the first attached pool containing the pool file lfn.

Files cannot be purged while they are open to a task. If the system cannot purge a file, it returns error messages to the dayfile of a batch job or to the terminal of an interactive user.

READCC - READ ALTERNATE CONTROL CARD FILE

The READCC control statement is valid only in a batch job as it is executed directly by the batch processor. It causes the batch processor to read control statements from an attached file other than the one containing the READCC control statement.

Figure 4-30 shows the READCC control statement format.

READCC, lfn.			
lfn	Name of file containing control statements. It cannot be INPUT.		

Figure 4-30. READCC Control Statement Format

The specified file must contain ASCII data in unstructured format and contain only control statements. Any valid control statement can appear, including images of other READCC statements. READCC can be nested at as many as 8 levels. Reading from the specified file terminates when an ASCII end-of-file (#1C) is encountered within the data. Control then returns to the statement following the appropriate READCC.

REQUEST - CREATE LOCAL FILE

The REQUEST control statement (refer to figure 4-31) can define a local mass storage file or request a tape file.

The first parameter on the REQUEST statement must be the file name. File length, if specified, must be the second parameter. The user must not specify tape parameters when requesting a mass storage file or mass storage parameters when defining a tape file.

No message is returned for successful completion.

The user can specify only one volume for the tape file; multivolume tape files cannot be requested. REQUEST cannot create a file if a local or attached permanent file with the same name exists.

REQUEST can ensure that a file is created on a particular pack. It can also determine whether a given pack has adequate space to hold a file of the required size; if not, the utility returns a fatal error code. This utility allocates mass storage for the file and controls whether the file space is contiguous at creation and whether the file can be extended.

The NOSEGMENT and NOEXTEND parameters control the continuity of the mass storage file. The interaction between the NOSEGMENT and NOEXTEND parameters is as follows:

Extendable File	Segmentable File	Result
No	No	One segment. File cannot be extended.
No	Yes	File created as one or two segments. File cannot be extended.
Yes	No	File created as one segment. Noncontiguous segments can be added.
Yes	Yes	File created as one or two segments. Noncontiguous segments can be added.

A retention period for the file is an installation option. The SWITCH control statement can be used to specify a particular number of days the file is to be retained on mass storage.

Format for Mass Storage Files

REQUEST,lfn,len,ACCESS=acs,MNR=mnr,MXR=mxr,PC=pc,RMK=rmk,RT=rt,NOEXTEND,NOSEGMENT,PACK=packid,SECURITY=lvl,TYPE=type.

Format for Tape Files

REQUEST,lfn,ACCESS=acs,MNR=mnr,MXR=mxr,PC=pc,RMK=rmk,RT=rt,DEVTYPE=dev,DENSITY=den,LB=lbl,OWNER=ownid,TPMODE=tpm,VSN=vsu.

lfn	File name; one to eight letters or digits beginning with a letter. lfn must be the first parameter specified.
ACCESS=acs	File access permission. If ACCESS=acs is omitted, read and write permission is granted for a mass storage file and read only permission for a tape file. R Read access. W Write access. RW or WR Read and write access.
MNR=mnr	Minimum record length in bytes. If MNR=mnr is omitted, the minimum length is 1 byte.
MXR=mxr	Maximum record length in bytes or, for F format records, the fixed record length. If MXR=mxr is omitted, no maximum length is set.

Figure 4-31. REQUEST Control Statement Format (Sheet 1 of 2)

PC=pc ASCII padding character. If PC=pc is omitted, blank is used as the padding character.

RMK=rmk ASCII character used as the record delimiter in R format files.

RT=rt Record type. If RT=rt is omitted, R format is assumed.

DEVTYPE=dev Device type. If DEVTYPE=dev is omitted, a mass storage file is defined.

MS Mass storage.

MT 7-track tape.

NT 9-track tape.

T7 7-track tape.

T9 9-track tape.

For mass storage files only:

/len Number of 512 word blocks to be allocated for the file; decimal or hexadecimal value from 1 through #FFFF. If /len is specified, it must immediately follow the file name (lfn). If /len is omitted, eight blocks are allocated.

NOEXTEND Indicates that the file cannot be extended. The file length cannot extend beyond that specified by /len. If NOEXTEND is omitted, the file can be extended up to a percentage of the original file length as set by an installation parameter (refer to File Space Allocation in section 2).

NOSEGMENT Indicates that the file must be contiguous when created. If NOSEGMENT is omitted, the file could be allocated two noncontinuous segments of file space.

PACK=packid Identifier of the pack on which the file is to be created. The identifier must be six characters; if fewer are specified, blanks are appended; if more than six are specified, the identifier is truncated. If PACK=packid is omitted, the system selects a pack.

SECURITY=lvl Security level of file (1 through 255). If SECURITY=lvl is omitted, the file security level is the same as that of the job.

TYPE=typ File type. If TYPE=typ is omitted, the system assumes the file is a physical data file.

C Virtual code file.

P Physical data file.

For tape files only:

DENSITY=den Tape density. If DENSITY=den is omitted, the system assumes 800 bpi.

LO or 200 200 bpi (7-track).

HI or 556 556 bpi (7-track).

HY or 800 800 bpi (7- or 9-track).

PE or 1600 1600 cpi (9-track).

LB=lbl Tape labels. If LB is omitted, the system assumes a labelled tape.

L Labeled tape.

U Unlabeled tape.

OWNER=ownid Owner identification from VOL1 label (one to 14 ANSI characters, left-justified, blank-filled). If OWNER=ownid is omitted, the system assumes the owner identification is blank.

TPMODE=tpm Tape format. If TPMODE=tpm is omitted, 8-bit ASCII code is assumed.

ASC 8-bit ASCII code (7- or 9-track (same as BIN)).

AS6 6-bit ASCII code (7-track).

BCD External BCD code (7-track).

BIN Unformatted binary (7- or 9-track).

VSN=vsu Volume serial number for VOL1 label (6 ANSI characters indicating the volume to be mounted). If VSN=vsu is omitted, the operator mounts an available tape.

Figure 4-31. REQUEST Control Statement Format (Sheet 2 of 2)

RERUN - SET RERUN STATUS

The RERUN control statement is valid only within a batch job as it is executed directly by the batch processor. It reverses the effects of a preceding NORERUN control statement. When a job's status is rerun, the batch input file is rerun from the beginning (at whatever priority it was running) when the system is brought up after system failure that terminated the batch processor.

RERUN format is shown in figure 4-32. Figure 4-19 shows RERUN use.

RERUN.

Figure 4-32. RERUN Control Statement Format

An installation parameter can override the rerun capability.

RESOURCE - SET JOB RESOURCE LIMITS

The RESOURCE control statement (refer to figure 4-33) can establish the time limit, job category, priority, working set size limit, and large page limit for a batch job.

The RESOURCE statement is optional; each of its parameters is also optional. If the statement or any of its parameters are omitted, the system uses default values.

If specified, the RESOURCE statement must be the first statement in the job. It must immediately follow the STORE card or the link station job statement within the batch deck.

Job categories are described under Resource Allocation in section 3. Except for the default job category, JDEFAULT, the user must ask installation personnel for the job categories for which he is validated. The effect of the priority specification is described under Job Scheduling in section 3.

RESOURCE, TL=t, JCAT=j, PRIORITY=p, WS=w, LP=lp.			
TL=t	Job time limit in system seconds [†] (decimal integer between 1 and 599 940). If the specified limit exceeds the maximum limit the installation specified for the job category, the job is held in the input queue until an operator command enables its execution or evicts the job.	WS=w	Maximum working set size in blocks (decimal integer). If the specified limit exceeds the maximum limit the installation specified for the job category, the job is held in the input queue until an operator command enables its execution or evicts the job.
JCAT=j	Job category indicated by an installation-defined mnemonic (1 to 8 letters or digits). If JCAT=j is omitted, the job is assigned to the JDEFAULT job category. The JDEFAULT category is valid for all users. Its maximum time limit is 1200 seconds, its default and maximum priority is 2, its maximum working set size is determined by an installation parameter, and its large page limit is 0.	LP=lp	Maximum number of large pages that can be assigned to the job (decimal integer). If the specified limit exceeds the maximum limit, the installation specified for the job category, the job is held in the input queue until an operator command enables its execution or evicts the job.
PRIORITY=p	Job priority, 1 (lowest) to 15 (highest). If the specified priority exceeds the maximum priority for the job category, the job priority is set at the maximum for its category.		If the large page limit, when multiplied by 128, exceeds the working set size limit, the job is aborted. If LP=lp is omitted, the large page limit is zero.

[†]A system second is one million STUs. If desired, an installation can substitute SBUs for system seconds as the time limit unit. The calculation of an STU or an SBU is described in volume 2.

Figure 4-33. RESOURCE Control Statement Format

RETURN - EVICT LOCAL FILES OR DETACH PERMANENT FILES

The RETURN control statement (refer to figure 4-34) can perform any one of the following functions.

- Release mass storage space allocated to one or more local files.
- Detach one or more permanent files from the job's suffix.
- End the assignment of one or more tape files to the job.

Files cannot be returned while they are open to a task. The file must be assigned or attached to this suffix before it can be returned. If the system cannot return a specified file, it returns an error message to the dayfile of a batch job or to the terminal of an interactive user. When the utility is called with the * parameter, all files are returned except any batch input file open to the batch processor. The job itself, however, could terminate abnormally if required files are not available.

RETURN, { Ifn-list }, UNLOAD=x.	
*	
Ifn-list	List of 1 through 16 private file names. Each name must be one through eight letters and digits, beginning with a letter (except drop file names).
*	Indicates the system should return all files assigned or attached to the job.
UNLOAD=x	Indicates whether the tapes should be unloaded.
	N Rewind, but do not unload file.
	Y Rewind and unload file.

Figure 4-34. RETURN Control Statement Format

ROUTE - SPECIFY FILE DISPOSITION

The ROUTE utility controls file disposition. Only local and attached permanent files can be routed. Tape files cannot be routed. It is required when a print or punch file is to be directed to a specific device.

The ROUTE control statement format is shown in figure 4-35. The first parameter must be the logical file name; all other parameters are optional and can appear in any order.

If the file is to be routed to an access station, only the first seven characters of the file are sent to the station as the file name.

SET - CHANGE MEMORY LIMITS

The SET control statement (refer to figure 4-36) can change the current memory limits (the working set size limit or the large page limit) for the job. The maximum memory limits for the job are specified on the job RESOURCE statement or by default values. Initially, the current memory limits are set to the maximum memory limits. The current memory limits cannot exceed the maximum memory limits, and the current working set size limit and the current large page limit must not conflict.

ROUTE,lfn,DC=dc, { SAVE } ,IC=ic,FID=fid,EC=ec,CM=cm,ST=st,TID=tid,OT=ot,REP=n,DI=di,
 DEF optional CYBER 200 Link parameters.

lfn	Logical file name of file to be routed.		
DC=dc	File disposition: Punch in format indicated by IC and EC parameters.		
SC	Scratch; that is, the file is to be destroyed. Default.		
PR	Print in format indicated by IC parameter.		
PU	For files routed to an access station, punch in format indicated by EC parameter. For files routed to a unit record station, punch in format of default disposition.		
IN	For files routed to a CYBER front-end, enter the file in the input queue. Cannot be used to route file to CYBER 200 input queue. The file is assumed to be an R format file containing ASCII data and no carriage control characters. IC=ic is ignored.		
P1	Print on 501 printer.	} Valid for CYBER 200 Link only.	
P2	Print on 580-12 printer.		
LR	Print on 580-12 printer.		
LS	Print on 580-16 printer.		
LT	Print on 580-20 printer.		
PF	For files routed to an access station, save as permanent file.		
DEF	Indicates that the file is not to be disposed of by this ROUTE call. (All other parameters of the call are processed and values retained in internal system tables.) The file will subsequently be disposed of by a ROUTE call that omits the DEF parameter. If the DEF parameter is omitted, the file is released to the appropriate queue at the time the utility executes and the job can no longer reference the file.		
SAVE	Indicates that a copy of the file is to be made and routed. The copy of the file is a local file with the name Q5ROUTE. SAVE is the default for permanent files.		
IC=ic	Internal characteristics of the file:		
AS	R format file containing ASCII data; and if DC=PR, the file has ANSI carriage control characters.		
PA	R format file containing ASCII data; and if DC=PR, the file has ASCII carriage control characters. Default.		
B1	R format file containing binary data. Required for files to output as binary punch files.		
FID=fid	First five characters of the file name while the file is in the output queue; must be one through five letters or digits beginning with a letter. The system adds two sequence characters as the sixth and seventh characters of the file name. The eighth character for files to be output as a unit record station is blank. If the file is SAVED, default is the file name.		
EC=ec	Punch or print file external characteristics:		
Punch:	26	O26 keypunch format	
	29	O29 keypunch format; default	
	80	80-column binary format	

Figure 4-35. ROUTE Control Statement Format (Sheet 1 of 2)

	Print:	B4	Print file on BCD 48-character print train.
		B6	Print file on BCD 64-character print train.
		A4	Print file on ASCII 48-character print train.
		A6	Print file on ASCII 64-character print train.
		A9	Print file on ASCII 95-character print train.
CM=cm	For files to be routed to an access station or CYBER 200 link station, conversion mode:		
	DI	CYBER front-end display 6-bit code (64-character set). Default.	
	EC	CYBER 6-bit extended display code (128-character set). (access station only.)	
	BI	Binary. No conversion.	
ST=st	Mainframe identifier of the system where the file is to be output: This is an installation-defined identifier. AST is reserved for the access station.		
	Logical identifier of the mainframe to which the file is routed. The installation defines the identifier for each mainframe in the configuration. If the CYBER 200 does not have a unit record station, the user need not specify the ST parameter when routing print files (DC=PR).		
TID=tid	Identifier of terminal to which file is to be returned. The identifier for the CYBER 200 Link consists of two letters or digits. TID=0, TID, or TID=any other unrecognizable identifier causes file to be routed to the central site. Not meaningful for the unit record station. The identifier for the access station consists of one to seven letters and digits.		
OT=ot	For files to be routed to an access station, origin type for terminal:		
	B	Local batch. Default.	
	E	Remote batch.	
DI=di	Optional CYBER 200 link parameters. See CYBER 200 Link Reference Manual. For files routed to an access station, one to eight alphanumeric characters.		
REP=n	Specifies a file repeat count from 0 through 99. Values beyond this range are set to 0. A zero value results in one copy of the specified file being routed. The repeat option is ignored if the user selects deferred routing. Each copy of the file to be routed is given a unique name composed of the first three characters of fid, if it is specified, or of lfn if fid is not specified, followed by a two-digit sequence number. If REP=rep is omitted, one copy of the file is routed.		

Figure 4-35. ROUTE Control Statement Format (Sheet 2 of 2)

SET,WS=w,LP=lp.			
WS=w	Current working set size limit in blocks (decimal integer). If the specified limit exceeds the maximum working set size limit for the job or if the specified limit is smaller than the current large page limit (multiplied by 128), the job is aborted.		
	If WS=w is omitted, the current working set size limit is not changed.		
LP=lp	Current large page limit (decimal integer). If the specified limit exceeds the maximum large page limit for the job, the job is aborted.		
	If LP=lp is omitted, the current large page limit is not changed.		

Figure 4-36. SET Control Statement Format

SWITCH - CHANGE FILE CHARACTERISTICS

The SWITCH control statement changes any of the following characteristics of a local or an attached permanent file: file name, type, access, and retention period, maximum and minimum record lengths, record type, padding character, and record delimiting character. For files that can be executed, this utility can also be used to indicate the length of the drop file that is to be created when the file is called for execution. The drop file length is placed in the executable file's minus page and in other system tables. A privileged user can change the drop file length of a public file.

SWITCH control statement format is shown in figure 4-37. The first parameter is required; nlfn, if specified, must appear as the second parameter. All other parameters are optional and can appear in any order.

SWITCH,olfn,nlfn,TYPE=typ,ACCESS=acs,RETENTION=days,DROP=dlen,MNR=mnr,MXR=mxr,RT=rt,PC=pc,RMK=rmk,SFO=sfo.	
olfn	Name of existing local or attached permanent file. olfn is required.
nlfn	New name of file (one through eight letters or digits beginning with a letter).
TYPE=type	New file type. C Virtual code file. P Physical data file.
ACCESS=acs	New file access permission. R Read access. W Write access.
RETENTION=days	Number of days the file is to be retained on mass storage (0 through 1023).
DROP=dlen	Number of 512-word blocks in drop file to be created when the file is executed (0 through #FFFF).
MNR=mnr	New minimum record length in bytes.
MXR=mxr	New maximum or fixed record length in bytes.
RT=rt	New record type. F ANSI fixed length. R Record mark delimited. U Undefined. W Control word.
PC=pc	New ASCII padding character.
RMK=rmk	New ASCII record mark character.
SFO=sfo	New SIL file organization. S Sequential.

Figure 4-37. SWITCH Control Statement Format

TV - SET THRESHOLD VALUE

The TV control statement (refer to figure 4-38) is valid only within a batch job. It can perform either of the following functions depending on whether a + follows the specified value.

- Set a threshold value to be compared to the return codes from succeeding job tasks (+ specified).
- Set a threshold value to be compared to the highest return code from preceding job tasks (+ omitted).

Each job task returns a code (its return code) to the batch processor upon completion of the task. The batch processor compares the return code to the current threshold value to determine if job processing should continue.

- If the return code is less than or equal to the current threshold value, the job continues with the next job task.
- If the return code is greater than the current threshold value, the batch processor searches for the next EXIT statement in the job. If it finds an EXIT statement, the threshold value is set at 255 and job processing continues with the statement following the EXIT statement. If it does not find an EXIT statement, the job terminates immediately.

The threshold value is not checked if the system aborts a task. In that case, the batch processor does search for an EXIT statement.

With a TV control statement, the user can also test the return codes of preceding job steps independent of the batch processor test. If the user omits the + following the value specified on the TV statement, the batch processor compares the specified value with the highest return code returned by a preceding job task to determine if job processing should continue.

Utilities described in this manual return only codes ERROR and FATAL. The code ERROR corresponds to return code 4; the code FATAL corresponds to return code 8.

The value a given task returns is established by a Q5TERM SIL call or a TERMINATE system message executed within the task.

TV,value+.	
value	Threshold value (0 through 255).
+	Indicates the specified value should be compared against the return codes of succeeding job tasks. If + is omitted, the specified value is compared against the highest return code from preceding job tasks.

Figure 4-38. TV Control Statement Format

Update is a utility used to maintain and manipulate a mass storage file containing images of coded punch cards or their equivalent. The utility provides the user with features that are a subset of the Update capabilities available under the NOS or NOS/BE operating systems. The Update card image file cannot be interchanged between these systems, however, since internal file structures differ between the operating systems.

Typical use of Update involves maintenance of a group of FORTRAN subroutines or assembly language routines. For convenience, the user often specifies each routine as a separate deck, so that one routine can be changed or extracted without affecting other routines in the file. Because each card image in the deck has its own Update-supplied sequence number, it can be referenced individually. A card can be deleted and replaced by two others, for example, in order to correct a routine or to increase its functions. At user request a deck can be extracted from the card image file in a format acceptable to a compiler or assembler and used as if it had been entered into the system as a punch deck. Once a source card is in the Update card image file, any physical punch card can be destroyed.

A source deck that is to be maintained through Update must be made a part of a special format file known as a program library. Creation of a program library is accomplished through Update itself. Subsequently, the program library can be changed on an Update correction run: new decks can be added, existing decks removed, or the contents of any deck changed.

The contents of a deck need only be images of coded cards. Update makes no assumptions about card contents. While programs are customary contents, they are not required contents and Update is equally applicable to a set of data cards or any other text.

The user controls Update operations through the parameters on the Update control statement and through a file containing directives and text. The directives are supplementary instructions for Update; the text is source cards to be made part of the Update card image file. Together, the directives and text are called the input stream.

EXAMPLES

An example of an Update creation run in which several FORTRAN routines become a program library with three decks is shown in figure 5-1. The Update control statement indicates a new library is to be created with the name MYDECKS. The input for Update is specified as the file INPUT, which is also the default file name when the I parameter is omitted.

The first directive encountered in figure 5-1 is *DECK ; therefore, Update recognizes a creation run and begins construction of a new program library. All cards following *DECK, up until the second *DECK directive, are written as a deck with the name MAIN. The first card is assigned the identifier MAIN.2, the next MAIN.3, and so forth. (The *DECK directive itself is also part of the library and has the identifier MAIN.1.)

```
STORE card
RESOURCE,TL=5.
UPDATE,I=INPUT,N=MYDECKS.
DEFINE,MYDECKS.
7/8/9
*DECK MAIN
  PROGRAM MAIN ...
  .
  .
  .
  END
*DECK SUBPROG
  SUBROUTINE SUB1 ...
  .
  .
  .
  SUBROUTINE SUB2 ...
  .
  .
  .
*DECK ERRPROG
  SUBROUTINE SUB3
  .
  .
  .
6/7/8/9
```

Figure 5-1. Typical Update Creation Run

A new deck, with card identifiers in the form SUBPROG.n, begins when Update encounters the second *DECK directive. In this example, the main program is in a deck with the same name as the program, two subroutines are in a deck with the name SUBPROG, and a third subroutine is in a deck with the name ERRPROG. At the end of the Update run, a program library exists with three decks.

The DEFINE control statement makes the local file MYDECKS a permanent file. The file MYDECKS remains in the system after job termination.

Figure 5-2 shows a correction run using the program library created in figure 5-1. This example adds a new card with text DIMENSION UP(50) near the beginning of subroutine SUB2. Notice that the location of the insertion is identified by the card identifier assigned within the deck SUBPROG and not by the subroutine name.

```

STORE card
RESOURCE,TL=5.
UPDATE (Q,P=MYDECKS)
FORTRAN (I=COMPILE)
LOAD.
GO.
7/8/9
*IDENT FIXIT
*INSERT SUBPROG. 89
      DIMENSION UP(50)
*COMPILE SUBPROG,MAIN
6/7/8/9

```

Figure 5-2. Typical Update Correction Run

The Update control statement in figure 5-2 identifies the existing program library with the P parameter. Since the Q parameter appears, it also instructs Update to operate in quick mode rather than full mode. When Update begins execution, it reads the next unexecuted record of the batch job, which is presumed to contain the input stream. The first card in this stream gives a name (FIXIT) to the corrections being made. The second card identifies the location at which the new card is to be added; namely, after the card with the identifier SUBPROG.89. Since the third card in the input stream does not correspond to a directive, it is considered a text card. Within the program library it becomes identified as FIXIT.1. The last card in the input stream instructs Update to write decks MAIN and SUBPROG to a file in a format suitable for input to the FORTRAN compiler. By default, in the absence of a C parameter on the Update control statement, the file name is COMPILE.

The FORTRAN compiler call in figure 5-2 names a file COMPILE as having the program to compile. Output of compilation is then loaded and executed with the LOAD and GO control statements.

Neither the FORTRAN call nor execution is required in figure 5-2. They are shown only to provide the programmer with a source listing of active cards in the deck with their card identifiers or to confirm proper program execution.

GENERAL PROCESSING

During execution, Update manipulates several files known as the input file, new program library, source file, old program library, compile file, and the listable output file. File operations depend on whether Update is performing a creation run or a correction run.

An Update run is defined as all operations with a program library that results from a single Update call. Any given run is a creation run or a correction run:

- A creation run constructs a program library. It is the original transfer of punch cards or card images into Update format.
- A correction run changes an existing program library. As a result of the run, a new program library might be generated; but the program library is new only in the sense that the changes are incorporated into the existing program library. All history information remains.

File names are specified by parameters of the Update control statement, which are summarized in table 5-1. Table 5-2 shows a summary of Update directives used during a run.

TABLE 5-1. SUMMARY OF UPDATE CALL PARAMETERS

Parameter	Function
C	Specify name of compile file.
D	Define compile file card image width excluding Update sequence information.
F	Select full update mode and source file and compile file contents.
I	Specify name of file with input stream.
L	Select listable output file contents.
N	Specify name of new program library file.
O	Specify name of listable output file; content is determined by L parameter.
P	Specify name of old program library file.
Q	Select quick update mode.
S	Specify name of source file; content includes common decks and is determined by mode.
T	Same as S, but omits common decks.
8	Define compile file card image width including Update sequence information.
*	Redefine master control character for directives.
/	Redefine control character for comments.

UPDATE MODE AND FILES

All files used by Update must reside on mass storage; all files created by Update reside on mass storage. Any of these files can be stored on magnetic tape, but the user is responsible for any transfers between tapes and mass storage devices. Default length of all of these files is #100 small pages. Any of the Update default files are opened and used if they exist as attached permanent files. If the files do not already exist, Update uses Q5GETFIL to create them as local files.

An intermediate processing file created by Update when a new program library is being created has the file name TEMNEWPL. Its default length is #400 small pages or the length of the new program library, whichever is larger.

TABLE 5-2. SUMMARY OF UDPATE DIRECTIVES

Directive Keyword Abbreviation	Directive Format	Use
*AF	*ADDFILE lfn,ident.seqnum	Read creation directives and text from named file and insert after card identified.
*CA	*CALL deck	Write common deck to compile file.
*CD	*COMDECK deck,NOPROP	Define common deck and propagation parameter.
*C	*COMPILE deck1,deck2,...,deckn	Write specified decks to compile file, source file, and new program library.
	*COMPILE deck1.deck2	Write inclusive range of decks to these files.
*DK	*DECK deck	Define deck to be included in program library.
*D	*DELETE ident1.seqnum,ident2.seqnum	Deactivate inclusive range of cards.
	*DELETE ident.seqnum	Deactivate specified card.
*ID	*IDENT idname,B=n,K=ident,U=ident	Define correction set, bias for seqnum, and whether specified correction sets must be known or unknown to process this set.
*I	*INSERT ident.seqnum	Write subsequent text cards after card identified.
*PD	*PURDECK deck1,deck2,..., deckn	Permanently remove specified decks from program library.
	*PURDECK deck1.deck2	Permanently remove inclusive range of decks.
*P	*PURGE idname1,idname2,...,idname3	Permanently remove specified correction sets from program library.
	*PURGE idname1.idname2	Permanently remove inclusive range of correction sets.
	PURGE idname,	Permanently remove specified correction set and all sets introduced after it.
*RD	*READ lfn	Read directives and text from specified file.
*Y	*YANK idname1,idname2,...,idnamen	Temporarily remove specified correction sets from program library.
	*YANK idname1.idname2	Temporarily remove inclusive range of correction sets.
*YD	*YANKDECK deck1,deck2,...,deckn	Temporarily deactivate decks specified.
*/	*/ comment	Copy text to listable output file.

The files that Update creates or uses are described below. An input file is always required; an old program library file is also required for a correction run. Each of these files has a default file name, but any other name can be specified through the appropriate parameter on the Update control statement.

The content of any compile file, source file, or new program library produced during a correction run is affected by the Update mode. The mode of an Update run is determined by a combination of the omission or specification of the F and Q parameters on the Update control statement.

Normal (selective), full, or quick Update mode is selected by:

<u>Parameter</u>	<u>Mode</u>
Both F and Q omitted	Normal selective mode in which the only decks processed are those modified or otherwise selected for processing.
F specified	Full mode in which all decks on the old program library are processed.
Q specified	Quick mode in which only decks specified on COMPILE directives are processed.
Both F and Q specified	Quick mode.

Input File

The input file contains the input stream. The input stream consists of directives that provide the details of Update processing and any new cards to be added to the program library. The file name is specified by the I parameter of the Update call; the default file name is INPUT.

New Program Library

The new program library is the file of card images and internal information in a special format that only Update can process. It contains a deck list of the names of all decks in the file and a directory of all correction sets introduced into the file. Each card is represented in a format that adds a card identifier and adds history and status information known as correction history bytes. Blanks are compressed out of the card image.

A new program library is an output file created by Update. Initially, it is generated on a creation run. For subsequent correction runs, the previous new program library is used as an input file and identified as the old program library; a new program library that incorporates the changes made during a correction run is then output from the correction

run. The file name is specified by the N parameter of the Update call; the default file name is NEWPL.

Source File

The source file is a file output during a correction run. It consists of card images that would allow regeneration of a new program library in resequenced format during a subsequent creation run. Only active cards and decks are part of the source file. The file name is specified by the S parameter of the Update call; the default file name is SOURCE. The content of the file is controlled by the T, F, and Q parameters. The user is responsible for routing the file to a punch or other output device.

Old Program Library

The old program library is the file generated as a new program library in a previous creation or correction run. It contains a record of changes made since the program library was created. It is required for any correction run. The file name is specified by the P parameter of the Update call; the default file name is OLDPL. The old program library is an R format file with blank compression.

Compile File

The compile file is an output file that contains a copy of a deck in the program library restored to a format that can be processed by a compiler or assembler. Only active cards in the deck are part of the compile file. The file name is specified by the C parameter of the Update call; the default file name is COMPILE. The content of the file is controlled by the directives and the F or Q parameters of the UPDATE call, with the D or 8 parameter selecting the number of columns in the image of each card. The compile file is an R format file with blank compression.

List File

The listable output file is the print file containing information for the user. It shows the card identifiers assigned by Update which the programmer must use to reference a card image in any future correction run. The file name is specified by the O parameter of the Update call; the default file name is OUTPUT. Content of the file is controlled by the L parameter, with options that can select a listing of directives processed, errors, comments, and a list of card images in the program library.

CREATION OF PROGRAM LIBRARY

A creation run exists when the first directive of the input file, other than a comment, is DECK or COMDECK. If the first directive is READ and the first directive of the file being read is DECK or COMDECK, a creation run also exists. Even if an old program library file is assigned to the job, Update ignores its existence and processes the run in creation mode.

Directives that can be used in a creation run are limited to:

- READ
- DECK
- COMDECK
- ADDFILE

In a creation run, each DECK or COMDECK directive defines a deck to be inserted into the program library under construction. Update decks can be one of two types, regular decks or common decks. They differ in that common decks can be called by name so that they are inserted into the text of another deck when the compile file is being generated; one copy of the common deck exists on storage, but multiple copies can be part of an output file.

In practice, the text written to a program library is often FORTRAN or assembly language routines in their punch card format. Update considers all cards to be a string of characters and takes no recognition of card contents. For convenience, a user often assigns a different Update deck name to each routine but there is no requirement to do so. Update divides text cards into decks following directive instructions.

The order of decks in the program library is controlled by the user; decks appear in the order in which they are found in the input stream. A common deck must precede any regular deck that might call the common deck.

All cards following a DECK or COMDECK directive, up until the next DECK or COMDECK directive, are considered to be part of the deck and each receives a unique sequence number. The directive defining the deck itself is assigned a sequence number 1. Any READ directive among the text cards causes Update to temporarily stop reading from the current input stream and to read from the specified file until an end-of-file is encountered; reading then resumes from the main input stream. Text cards read from the file specified by READ are numbered as if they were part of the original input text.

CARD IDENTIFICATION

The image of each card stored in a deck contains information known as correction history bytes. This information, generated by Update, maintains the history and status of a card and is the means by which Update can reverse status. Deletion of a card, for example, is accomplished by the addition of a correction history byte to the card image rather than a physical deletion of the image. Consequently, the card can be reactivated at some later time. Only purge operations are irreversible.

A DECK or COMDECK directive is written to the program library as part of the deck text. Consequently, these directives can be referenced just as any other card in the text. Deactivating a DECK directive, for example, has the effect of making its following text a part of the deck that precedes it in the library.

Update recognizes one full form and two short forms of card identifiers. The full form card identifier is:

ident.seqnum

ident. One through eight character name of a correction set or deck. A period terminates the ident name.

seqnum Decimal ordinal (1 through 65 535) representing the sequence number of the card within the correction set or deck. Any character other than 0 through 9 terminates the sequence number.

The two short forms of card identifiers can be used on INSERT or DELETE directives. The short forms are expanded as follows:

seqnum Expands to idname.seqnum where idname is a correction set identifier, whether or not it is also a deck name.

.seqnum Expands to dname.seqnum where dname is a deck name.

In the short form, idname is assumed to be the last explicitly named ident given on an INSERT or DELETE directive, whether or not it is a deck name. The dname is assumed to be the last explicitly named ident given on an INSERT or DELETE directive that is known to be a deck name. Both of these default idents are originally set to YANK\$\$\$ so the first directive using a card identifier must use the full form to reset the default.

All deck names are also idents (but all idents are not decks). Thus, if EXAMPLE is the deck name last used, and there is no subsequent explicit reference to a correction set identifier, then both .281 and 281 expand to EXAMPLE.281 as the identifier. If there is an explicit reference to a correction set identifier after the explicit reference to the deck name, then 281 would expand to the correction set identifier, while .281 would expand to EXAMPLE.281 as the identifier.

Figure 5-3 shows differences in identifier expansion depending on the order of directive records, assuming A is a deck name and B is a correction set identifier on an Update old program library.

*ID C		
*INSERT A.2	data card	
*INSERT B.1	data card	
*D 2, 3		expands to *DELETE B.2, B.3
*D 4, .5		expands to *DELETE B.4, A.5
*D .7, 5		expands to *DELETE A.7, B.5
*D .9, .10		expands to *DELETE A.9, A.10
whereas:		
*ID D		
*INSERT B.1	data card	
*INSERT A.2	data card	
*D 2, 3		expands to *DELETE A.2, A.3
*D 4, .5		expands to *DELETE A.4, A.5
*D .7, 5		expands to *DELETE A.7, A.5
*D .9, .10		expands to *DELETE A.9, A.10

Figure 5-3. Card Identifier Expansion

CORRECTION RUN.

A correction run, which is the most common use of Update, introduces changes into the existing program library. Update recognizes a correction run, as opposed to a creation run, under either of the following circumstances:

- The first directive, other than a comment, is IDENT.
- The first directive, other than a comment, is READ or ADDFILE and the first directive on the alternate file is IDENT (in the case of READ) or DECK or COMDECK (in the case of ADDFILE).

All directives can be used during a correction run.

The IDENT directive establishes a name for the correction set. Any cards inserted into the library are sequenced within this name. On subsequent correction runs, individual cards in the correction set can be referenced by sequence number. The entire correction set can also be referenced as a whole.

When a new program library is being generated, all corrections must be part of a correction set with the exception of PURGE, PURDECK, and ADDFILE. That is, IDENT must be the first directive other than a comment. If READ is the first directive, the alternate input file must have IDENT as its first directive. If a new program library is not being generated (that is, routines are being extracted, but no changes made), directives can appear without a correction set identifier.

Four directives need not be part of a correction set. They are directives PURGE, PURDECK, and ADDFILE (which cause the current set to be terminated) and COMPILE. The COMPILE directive, as with the comment directive, can appear anywhere within or outside a correction set. It is not processed until all corrections have been made.

More than one correction set can be introduced during a single run. The correction set established by the first IDENT directive remains in effect until Update either encounters another IDENT directive or encounters a PURGE, PURDECK, or ADDFILE directive. The subsequent IDENT directive establishes a second correction set name.

A correction run can include the addition of new decks to the program library when a new program library is created. Decks to be added are identified by a DECK or COMDECK directive following an INSERT, DELETE, or ADDFILE directive.

Deck List and Directory Order

Update maintains a list of all decks in the program library known as a deck list. The order of entries in the deck list is under user control; original deck list entries correspond to the order in which decks are written during the creation run. Subsequent additions of decks are made at the location specified by the user with a preceding INSERT, DELETE, or ADDFILE directive; or, if the directive

preceding DECK or COMDECK is IDENT, at the end of the current library.

The location of an entry in the deck list is significant in terms of parameters for PURDECK, YANKDECK, and COMPILE directives in which a range of decks is referenced. The order of names in a range reference must be the same as the order in the deck list. The decks named and all those between are then processed in accordance with the directive. An error exists if they are in reverse order.

Similarly, as each correction set is introduced into the program library, Update creates an entry in an internal directory in chronological sequence. The location of an entry in the directory is significant in terms of parameters for PURGE and YANK directives in which a range of correction sets is referenced. The order of reference must be the same as the order of the directory. The identified correction sets and all those between are processed in accordance with the directive. An error exists when a correction set range is not referenced in the order the sets were introduced into the library.

PURGE and YANK Directives

The two purge directives are PURGE and PURDECK. PURGE operates on all cards identified by correction set name, while PURDECK operates on all cards within an identified deck. (Introduction of a new deck on a creation run must be made as part of a correction set, but that addition usually is the only change within that correction set.) The two yank directives are YANK and YANKDECK. As with the purge directives, the former operates with correction sets and the latter with decks.

The purge directives differ from the yank directives in that yank operations are temporary. Cards yanked from the program library are temporarily deactivated. They can be reactivated by a subsequent yank of the yank directive that inactivated the cards.

In contrast, any change made to a program library through a purge directive is permanent. A reversal of a purge operation is possible only through the reintroduction of the cards into the library as if they had not previously existed.

Since the YANK directive itself must be introduced as part of a correction set, a future correction set that references the correction set containing the YANK reactivates the original correction set. For instance:

To inactivate all cards added by IDENT PSR003:

*IDENT TAKEOUT
*YANK PSR003

To reactivate the same cards:

*IDENT PUTBACK
*YANK TAKEOUT

Overlapping correction messages might be produced as a result of these procedures.

Update stores all YANK directives in a deck with identifier YANK\$\$\$. Individual cards in the deck, but not the entire deck, can be referenced. An alternative to the two directives above that yanked identifier TAKEOUT, is:

***PURGE YANK\$\$\$.TAKEOUT**

Once a card or an entire correction set has been yanked, it cannot again be referenced except for a reactivation request. That is, a yanked card cannot be referenced on DELETE or INSERT.

Overlapping Corrections

Update can detect four overlapping correction situations. When any of these types is detected, Update prints the offending line with the words TP.n OVLP appended on the far right.

<u>Type</u>	<u>Meaning</u>
Type 1	Two or more modifications are made to one card by a single correction set.
Type 2	A modification attempts to activate an already active card.
Type 3	A modification attempts to deactivate an already inactive card.
Type 4	A card is inserted after a card which was inactive on the OLDPL.

Detection of an overlap does not necessarily indicate a user error. Overlap messages are advisory, and they point to conditions in which the probability of error is greater than normal.

Type TP.2 and TP.3 are detected by comparing existing correction history bytes with those to be added. Complex operations involving YANK and PURGE might generate these overlap messages even though no overlap occurs.

Modifications for each correction set are performed by Update in the order in which sets are introduced. The order is irrelevant if no correction is dependent on another. If a dependent relationship exists, however, order is of paramount importance.

UPDATE DIRECTIVES

Directives are instructions for Update to follow in creating its output files. A directive must begin in column 1 with the master control character. Each directive has both a full keyword and an abbreviated keyword, as shown in table 5-2.

General format is:

***keyword p-list**

***** Master control character which distinguishes a directive from a text card. Must appear in column 1. This character can be changed through the ***=c** parameter of the Update control statement.

keyword Name of one of the Update directives or an abbreviation for a directive. No blanks can occur between the master control character and the keyword; a comma or blank terminates the keyword.

p-list Parameters identifying decks, cards, or files. Multiple blanks can appear between the keyword and parameters. Parameters in the list are separated by commas; embedded blanks cannot appear in the list.

Notice that several parameters contain a period as part of a single parameter.

No terminator appears at the end of a directive.

The master control character is recorded in the program library. For a correction run, the master control character should match the character used when the program library was created; if the characters do not match, Update uses the character stored as part of the program library.

Any card in the input stream that cannot be recognized as a directive or as a comment is assumed to be text.

Directives are described in alphabetical order, following.

ADDFILE DIRECTIVE

The ADDFILE directive causes Update to add a file of new decks to the new program library. It differs from the READ directive in that contents of the specified file is limited to those that add decks. The first card of the specified file must be a DECK or COMDECK directive. No directives other than comments, DECK, or COMDECK can appear in the file.

The ADDFILE directive format is shown in figure 5-4. If only one parameter appears, it is assumed to be lfn.

*ADDFILE lfn,ident.seqnum	
lfn	Name of file from which decks are to be added. Default is the file specified by the I parameter of the Update call.
ident.seqnum	Identifier of card after which decks are to be placed on the program library. If omitted, the addition is made at the end of the program library.

Figure 5-4. ADDFILE Directive Format

When the specified file is not INPUT, Update reads directives and text cards until an end-of-file (#1C) is encountered. Update then returns to the file specified by the I parameter of the Update call and continues processing the main input stream. When the file specified on the ADDFILE directive is INPUT, however, Update reads directive and text cards only until either an end-of-file or an Update directive other than DECK, COMDECK, or CALL is encountered.

An ADDFILE directive cannot appear among directives read from a file specified by a READ directive; otherwise, it can appear anywhere in the input stream, but its appearance terminates the current correction set.

CALL DIRECTIVE

The CALL directive causes Update to write the text of a previously encountered common deck onto the compile file. The directive itself is stored as part of a deck and can be referenced by its sequence number. It is effective only within a deck.

The CALL directive format is shown in figure 5-5.

*CALL deck	
deck	Name of an existing common deck to be written to the compile file.

Figure 5-5. CALL Directive Format

Neither the CALL directive nor the COMDECK directive, which defined the deck, becomes part of the compile file.

Common decks can call other common decks, but a common deck must not call itself or a deck that contains a call to the common deck.

COMDECK DIRECTIVE

The COMDECK directive establishes a common deck that can be called from other decks as they are being written to the compile file. It is one of the two directives that establishes the existence of a creation run. The directive can be used in any correction run to add a common deck to a particular location in the program library.

The COMDECK directive format is shown in figure 5-6.

*COMDECK deck,NOPROP	
deck	Name of common deck being added. Must be one through eight characters A through Z, 0 through 9, or + - / * () \$ = _ . Must not duplicate the name of an existing deck.
NOPROP	Indicator that decks calling this common deck are not to be considered as modified when the common deck itself is modified; that is, the effects of common deck changes are not to be propagated during a normal Update mode. Optional.

Figure 5-6. COMDECK Directive Format

The COMDECK directive itself is part of the program library and has a sequence number of 1 within the name established by the directive. For a creation run, the deck order in the input stream determines the location of the common deck in the program library. For a correction run, the location in the program library is determined by the preceding INSERT directive or by the location resulting from a preceding DELETE or ADDFILE. Common decks need not appear first on the program library, but they must appear before any decks from which they are called during a creation run.

The NOPROP parameter of the COMDECK directive that created a common deck determines whether a deck calling a corrected common deck will also be considered corrected.

COMPILE DIRECTIVE

The COMPILE directive affects the decks to be written to the compile file and any new program library or source file during normal or quick Update mode. The directive is ignored during a full Update.

Normal mode	Decks specified on COMPILE directives and corrected decks are written to the compile file
Quick mode	Decks specified on COMPILE directives and any common decks they call are written to the compile file.

The COMPILE directive format depends on whether decks to be written are specified individually by name or are specified as a range of deck names, as shown in figure 5-7.

A. Compile listed decks	
*COMPILE deck1,deck2, ...,deckn	
deck	Name of deck to be written to the compile file, new program library file, and source file.
B. Compile range of decks	
*COMPILE deck1.deck2	
deck1.deck2	Names of first and last decks in range, inclusive, to be written to the compile file. The name of deck1 must appear in the old program library deck list before deck2.

Figure 5-7. COMPILE Directive Format

Decks are always written in the order that the decks exist on the old program library.

COMPILE directives can appear anywhere within the input stream. They do not affect the current correction set name.

The compile directive also affects the contents of any new program library and source file, as shown in table 5-3.

TABLE 5-3. FILE CONTENTS AND UPDATE MODE

File	Normal Mode Contents	Full Mode (F) Contents	Quick Mode (Q) Contents
New program library	All regular and common decks† after corrections made in sequence of old program library.	Same as normal mode source file.	Decks specified on COMPILE directives, any common decks† they call, and any common decks encountered on old program library prior to all decks of COMPILE.
Compile File	All decks corrected or listed on COMPILE directives, and any deck calling a corrected common deck (unless NOPROP specified on COMDECK).	All active decks on old program library.	All decks on COMPILE directives and any common decks† they call.
Source File	All currently active DECK, COMDECK, and CALL directives and active text required to recreate library.	Same as normal mode source file.	Currently active cards required to create new program library resulting from quick mode.
†T parameter excludes common decks.			

DECK DIRECTIVE

The DECK directive establishes a deck in the program library. It is one of the two directives that establishes the existence of a creation run. The directive also can be used in any correction run to add a deck to the location indicated by a preceding ADDFILE directive.

The DECK directive format is shown in figure 5-8.

Each deck must have a unique name within the program library.

The DECK directive itself is part of the program library and has a sequence number of 1 within the name established by the directive.

*DECK deck	
deck	Name of deck. Must be one through eight characters A through Z, 0 through 9, or + - / * () \$ = __. Must not duplicate the name of any other deck in program library.

Figure 5-8. DECK Directive Format

DELETE DIRECTIVE

The DELETE directive deactivates a card or group of cards and optionally adds text cards following the directive. A deactivated card remains on the library and retains its sequencing. It can be referenced just as if it were not deactivated. A deactivated card is not written to any compile file or source file, however.

The DELETE directive format depends on whether cards to be deactivated are specified by card identifier or by a range of cards, as shown in figure 5-9.

A. Delete specified card

***DELETE ident.seqnum**

ident.seqnum Card identifier for single card to be deleted.

B. Delete range of cards

***DELETE ident1.seqnum,ident2.seqnum**

ident1.seqnum, Card identifiers of
ident2.seqnum first and last cards,
inclusive, in sequence of cards to be deleted. Card ident1.seqnum must appear before ident2.seqnum in the existing library. The range can include cards in a deactivated state.

Figure 5-9. DELETE Directive Format

IDENT DIRECTIVE

The IDENT directive establishes the name for the set of corrections being made. Cards added in this correction set are sequenced within the name specified. Any card whose status is changed by this set receives a correction history byte that references the name from IDENT. All correction set names must be unique.

IDENT directive format is as shown in figure 5-10.

*IDENT ident,B=num,K=ident,U=ident	
ident	Name to be assigned to this correction set. Must be one through eight characters A through Z, 0 through 9, or + - / * () \$ = _ . Must not duplicate the name of another correction set or deck.
B=num	Bias to be added to sequence numbers within deck.
K=ident	Indicator that specified correction set name must exist in the directory of the library before corrections can be made.
U=ident	Indicator that specified correction set name must not exist in the directory of the library.

Figure 5-10. IDENT Directive Format

The B, K, and U parameters can appear in any order. More than one K or U parameter can be specified; in this instance, all correction set names specified must meet the criteria before the correction set is processed. If the criteria of these parameters is not met, Update skips the correction set and resumes processing with the next IDENT, PURGE, PURDECK, or ADDFILE directive.

INSERT DIRECTIVE

The INSERT directive adds text cards following it to the program library at the location specified.

The INSERT directive format is shown in figure 5-11.

New cards receive card identifiers established by the correction set name of the preceding IDENT directive.

*INSERT ident.seqnum	
ident.seqnum	Card identifier of card after which the insertion is to be made.

Figure 5-11. INSERT Directive Format

PURDECK DIRECTIVE

The PURDECK directive permanently removes a deck or group of decks from the program library. Every card in the deck is purged, regardless of what correction set it might belong to. Purging, unlike yanking, cannot be rescinded.

The PURDECK directive format depends on whether decks to be purged are specified individually by deck name or by a range of deck names, as shown in figure 5-12.

A. Purge decks listed

***PURDECK deck1,deck2, . . . ,deckn**

deck Name of deck to be purged. Names can appear in any order.

B. Purge range of decks

***PURDECK deck1.deck2**

deck1.deck2 Names of first and last decks, inclusive, to be purged. Names must appear in the relative order in which decks exist in the deck list.

Figure 5-12. PURDECK Directive Format

A PURDECK directive can appear anywhere in the input stream, but its appearance terminates the current correction set. Any directive following PURDECK must be another purge directive or a directive that institutes another correction set. The deck YANK\$\$\$ cannot be purged.

The name of the purged deck can be reused as a deck name. It can be used as a new correction set identifier only if it does not already exist in the directory.

PURGE DIRECTIVE

The PURGE directive permanently removes a correction set or group of correction sets from the program library. Every card in the correction set is purged, regardless of its status as active or inactive. Purging, unlike yanking, cannot be rescinded.

The PURGE directive format depends on whether correction sets to be purged are specified individually by correction set name, by a range of correction set names, or by relative time of introduction into the program library, as shown in figure 5-13.

A PURGE directive can appear anywhere in the input stream, but it terminates the current correction set. Any directive following PURGE must begin a new correction set.

READ DIRECTIVE

The READ directive causes Update to temporarily stop reading the current input stream and to begin reading an input stream from the file specified on the READ directive. READ differs from ADDFILE in that the content of the file specified by READ is not restricted except to prohibit the appearance of either another READ directive or an ADDFILE directive. Update reads from the specified file until an end-of-file (#1C) is encountered. Processing then continues with the main input stream.

A. Purge listed correction sets

*PURGE ident1,ident2, . . . ,identn

ident Identifier of a correction set to be purged. Identifiers can appear in any order.

B. Purge range of correction sets

*PURGE ident1.ident2

ident1.ident2 Identifiers of first and last correction sets, inclusive, to be purged. Identifiers must appear in the relative order in which the correction sets were introduced into the program library; that is, they must appear in the order they exist in the directory.

C. Purge later correction sets

PURGE ident,

ident Identifier of correction set to be purged along with all correction sets introduced after the specified correction set.

* Indicator that the program library is to return to an earlier level. Intervening purge directives prevent complete return.

Figure 5-13. PURGE Directive Format

*READ lfn

lfn Name of alternate file containing input stream.

Figure 5-14. READ Directive Format

YANK DIRECTIVE

The YANK directive temporarily removes a correction set or group of correction sets from the program library. Cards activated by the correction set are deactivated; cards deactivated by the correction set are reactivated. YANK differs from purge in several respects: YANK must be part of a correction set; it does not terminate the current correction set; its effects can be rescinded.

The YANK directive format depends on whether correction sets to be yanked are specified individually by correction set name or by a range of correction set names, as shown in figure 5-15.

Update places the YANK directive in the YANK\$\$\$ deck. If a correction has been yanked, it is ignored during compile file or source file generation.

A. Yank listed correction sets

*YANK ident1,ident2, . . . ,identn

ident Identifier of a correction set to be yanked. Identifiers can appear in any order.

B. Yank range of correction sets

*YANK ident1.ident2

ident1.ident2 Identifiers of first and last correction sets, inclusive, to be yanked. Identifiers must appear in the relative order in which the correction sets were introduced into the program library; that is, they must appear in the order they exist in the directory.

Figure 5-15. YANK Directive Format

YANKDECK DIRECTIVE

The YANKDECK directive temporarily deactivates all cards within the decks specified. All cards are deactivated, regardless of the correction set to which they belong. YANKDECK differs from PURDECK in several respects: YANKDECK must be part of a correction set; it does not terminate the current correction set; its effects can be rescinded.

The YANKDECK directive format is shown in figure 5-16.

*YANKDECK deck1,deck2, . . . ,deckn

deck Name of deck to be yanked. Names can appear in any order.

Figure 5-16. YANKDECK Directive Format

The deck YANK\$\$\$ cannot be deactivated as a whole. Individual YANK directives within this deck can be yanked by a YANK directive, however.

/COMMENT DIRECTIVE

The / comment directive introduces a comment into the listable output file. Update ignores this card except to copy it to the output file. A comment can appear at any place in the input stream.

The / comment directive format is shown in figure 5-17. The slash must appear in column 2. Column 3 must be a comma or a blank. The slash can be redefined as another character through the /c parameter of the Update call.

*/ comment

Figure 5-17. / Comment Directive Format

UPDATE CONTROL STATEMENT

The format of the control statement that calls Update to execution is shown in figure 5-18. All parameters are optional and can appear in any order. A comma must separate parameters.

Update, $\left\{ \begin{array}{c} C \\ C=file \end{array} \right\}$, D, F, $\left\{ \begin{array}{c} I \\ I=lf n \end{array} \right\}$, L=opt, $\left\{ \begin{array}{c} N \\ N=lf n/\#nnn \end{array} \right\}$, $\left\{ \begin{array}{c} O \\ O=lf n/\#nnn \end{array} \right\}$, $\left\{ \begin{array}{c} P \\ P=lf n \end{array} \right\}$, Q, $\left\{ \begin{array}{c} S \\ S=lf n/\#nnn \end{array} \right\}$,
 $\left\{ \begin{array}{c} T \\ T=lf n/\#nnn \end{array} \right\}$, 8, *=c, /c.

C Compile file name. The content of the compile file is determined by the Update mode.

omitted Decks are written to the file named COMPILE.
or C

C=lf n Decks are written to file named lf n. File length is #100 small pages or the number
or of pages specified by nnn.
C=lf n/#nnn

C=PUNCH Decks are written to file named PUNCH. The D and 8 parameters are implied.

C=0 Compile file suppressed.

D Data width on compile file excluding Update sequence identifiers.

omitted 72 columns of data.

D 80 columns of data.

F Full Update mode.

omitted Normal selective Update mode, as long as Q is not specified. The compile file contains only those decks corrected in this run or otherwise specified on COMPILE directives.

F Full Update mode. The compile file contains all active decks in the program library.

I Input stream file name.

omitted Directives and text are on the file named INPUT.
or I

I=lf n Directives and text are on file named lf n.

L Listable output options to be written to file named with the O parameter.

omitted For a creation run, options A, 1, and 2.

For a correction run, options A, 1, 2, 3, and 4.

L=c...c Each character in string c...c selects one of the following options.

A Error decks, correction set identifiers, common decks, and decks written to the compile file are listed.

F Full listing.

0 The character 0 overrides any other options specified and suppresses the entire listing.

Figure 5-18. Update Control Statement Format (Sheet 1 of 3)

	1	Suppress deck name list, identifier list, and continuous commentary when L=1 is specified. List errors if this option is selected by omission of the L parameter.
	2	List directives with ***** preceding each directive with valid format. Each IDENT directive begins a new page of the listing.
	3	Comment on each card changed. Comments include the deck name, card image, card identifier and sequence number, and an indicator of action taken for that card: I Card added. A Inactive card reactivated. D Active card deactivated. P Card purged. If the card was active, ACTIVE also appears.
	4	List cards of input stream established by directives. Cards read as a result of a READ directive are identified to the right with the file name; cards inserted as a result of an ADDFILE directive are listed only when option 4 is explicitly selected. *ERROR* accompanies any cards in error.
	9	List all active and inactive cards with status: I Inactive. A Active. Option 3 overrides option 9.
	L=0	Suppress all listings.
	N	New program library file name.
	omitted	In a correction run, suppress new program library generation. In a creation run, write a new program library on the file named NEWPL.
	N	Write new program library on file named NEWPL.
	N=lfm or N=lfm/nnn	Write new program library on file named lfm. File length is #100 small pages or the number of pages specified by nnn.
O	Listable output file name.	
	omitted or O	Write list output to file named OUTPUT.
	O=lfm or O=lfm/nnn	Write list output to file named lfm. File length is #100 small pages or the number of pages specified by nnn.
P	Old program library file name.	
	The P parameter is valid only for a correction run.	
	omitted or P	Old program library resides on file named OLDPL.
Q	P=lfm	Old program library resides on file named lfm.
	Quick Update mode. The source file and the new program library are described in table 5-3.	
	omitted	When F is also omitted, normal selective Update mode.
Q	Only those decks specified on COMPILE directives are processed. Corrections to decks not specified on COMPILE are ignored, except for ADDFILE. The compile file contains only decks referenced on COMPILE directives and the common decks they call.	
	The Q parameter takes precedence when both F and Q are specified.	

Figure 5-18. Update Control Statement Format (Sheet 2 of 3)

S Source file name. The content of this file is determined by the Update mode.

omitted Suppress source output file unless it is selected by the T parameter.

S Source output file to be written on file named SOURCE.

S=lfm Source output file to be written on file named lfm. File length is #100 small
or
S=lfm/#nnn pages or the number of pages specified by nnn.

T Omit common decks from source file. The content of the source file is determined by the Update mode,
with the T parameter excluding common decks.

omitted Suppress source output unless it is selected by the S parameter.

T Source output file to be written on file named SOURCE, with common decks excluded.

T=lfm Source output file to be written on file named lfm, with common decks excluded.
or
T=lfm/#nnn File length is #100 small pages or the number of pages specified by nnn.

The T parameter takes precedence over the S parameter.

8 Card image width on compile file including Update sequence identifiers.

omitted 90-column card image, which preserves columns 73 through 80 of original card.

8 80-column card image, with Update sequence information in columns 73 through 80.

***** Master control character for directives.

omitted * is the first character of each directive.

***=c** c is the first character of each directive for this Update run. c can be any character A
through Z, 0 through 9, or + - * / \$ or =. If the character specified for a correction run is not
the same as the character used when the old program library was created, the old program
library character is used.

/ Comment control character in column 2

omitted Comment control character is /.

/=c c is the comment control character. c can be any character A through Z, 0 through 9, or + - *
/ \$ or =.

Figure 5-18. Update Control Statement Format (Sheet 3 of 3)

DEBUG, LOOK, and DUMP are utilities for testing and debugging a correctly compiled or assembled program that executes unsatisfactorily. Also available for debugging purposes is the DEBUG parameter of the LOAD control statement in section 4. These utilities can be executed either interactively or in batch mode.

Differences among these three utilities include:

- DEBUG displays or alters the contents of selected locations during program execution. It is valid only with controllee files.
- LOOK displays or alters the contents of selected locations in any type of file. It can be used with controllee files or data files. Its most common use is through an interactive terminal.
- DUMP displays a preselected set of elements from a dump file.

Both LOOK and DEBUG use a set of directives supplied by a programmer to receive detailed control information. A batch job must have the directives on a file available to the job.

An interactive user can enter directives interactively and receive output as it is generated in response to the directive. Output from most directives is returned to the terminal; some directives can specify a file to receive output. When the utility is ready for another directive, the character ? appears at the terminal. Directives must be entered on a single line.

Typical use of the debugging utilities involves using LOOK to edit a FORTRAN source program interactively until the program compiles successfully; executing the compiled program and possibly receiving a dump on a fatal error condition, or else possibly forcing such a dump by making a DUMP request; using DEBUG to observe intermediate program values during reexecution of the program under DEBUG control; and subsequently using DEBUG or LOOK to modify the program until it executes satisfactorily.

DEBUG

Through DEBUG the user can set breakpoints in a program and then issue EXECUTE and CONTINUE directives to step through the execution of the program from one breakpoint to the next. At each breakpoint, current values of variables in the program can, for example, be dumped. DEBUG can also be used to modify, display, and dump user registers and areas in virtual memory designated by hexadecimal addresses.

A FORTRAN program being executed under DEBUG must have been compiled without the S compile option if symbolic addresses - labels, names, line numbers - are to be used in the DEBUG directives. DEBUG executes entirely within the user's virtual space, starting at hexadecimal virtual bit address #7FFF00000000 and extending upwards; therefore, the program being debugged must not use or reference this area.

After the DEBUG control statement is issued, DEBUG remains in execution until an EXECUTE, STEP, or CONTINUE directive causes it to relinquish control to the user program. Control does not return to DEBUG until a user-specified breakpoint is reached during execution. When the user program terminates, DEBUG terminates also; more DEBUG directives can be processed only after another DEBUG control statement has been issued.

DEBUG CONTROL STATEMENT

The control statement that initiates execution of DEBUG is shown in figure 6-1. The parameter fname must always be the first parameter; but the order of the I= and O= parameters can be reversed.

DEBUG,fname,I=iname,O=oname/olen.	
fname	Name of the existing permanent or local file that is to be the controllee file for DEBUG. It must be a virtual code file produced by the LOAD utility.
I=iname	For batch mode only, a file containing the DEBUG directives. If I=iname is omitted, directives are read from INPUT.
O=oname/olen	For batch mode, the file to which all DEBUG output is written; for interactive mode, the file to which data generated by the SNAP command is written. olen is the length of the output file in small pages.
The default file is OUTPUT. The default file length is 100 small pages.	

Figure 6-1. DEBUG Control Statement Format

The following are sample DEBUG control statements.

- DEBUG(MYCTEE)
- DEBUG(MYCTEE,I=MYINP,O=MYOUT/#2C3)
- DEBUG(MYCTEE,O=MYOUT,I=MYINP)

DEBUG DIRECTIVE

The general format of each DEBUG directive is as follows.

Parameters are positional and can be separated from each other and the directive name by either a blank or a comma. A null parameter must be indicated by commas delimiting its position.

directive,parameter-set

DEBUG directives are listed in alphabetical order in figure 6-2. The directive descriptions are grouped according to a common function.

Examples of DEBUG directives are shown in figure 6-3.

ASCII	Enter data in ASCII form.	EXECUTE	Begin execution of user program at a specified location.
BACK	Display the data preceding the last display location.	FLOAT	Enter data in floating point.
BKPT or BKPTR	Set or remove breakpoints.	HEX	Enter half-word hexadecimal data.
CONTINUE	Continue execution from the last user breakpoint.	IDISPLAY	Display the data contained at the address found at the specified location.
DDECIMAL	Display data in hexadecimal and decimal.	IDREG	Display the data found at the address specified in the given register.
DECIMAL	Enter data in decimal form.	ROLL	Display the data following the last display location.
DFLOAT	Display data in hexadecimal and floating point.	SNAP	Dump to an output file.
DISPLAY	Display data in hexadecimal and ASCII.	STAT	Provide status information such as breakpoints set, last routine referenced, and last directive issued.
DREG	Display register contents in hexadecimal.	STEP	Step through execution of user code one or more instructions at a time.
END	Terminate execution of both DEBUG and user program.		
EREG	Enter half-word hexadecimal data into a register.		

Figure 6-2. DEBUG Directives

DI SUBR=500+4,5	If this is the first directive entered under DEBUG or if the last type referenced (if referenced at all) was S, this directive displays five words starting at four words after the location labeled 500 in module SUBR. If SUBR does not contain a label 500, the following message is displayed.	BKPT 111/L	Sets a breakpoint at the location corresponding to source line number 111 in the current module. An error message is displayed if the current module is not at least 111 lines long.
	NO SUCH SYMBOL.	DE/H 4A0/X-10	Places -10 (decimal) in the halfword at 4A0 from the beginning of the current module.
HEX SUBR=500/X 1000C 880	Enters two half words of hexadecimal data at bit address 500 in module SUBR.	DI 0=C840/X	Displays four words beginning at absolute virtual address C840.

Figure 6-3. Sample DEBUG Directives

Dump or Display Directives

The user can display the contents of up to 16 words of virtual memory by entering one of the following:

DISPLAY, [name=] location [/type] [+offset] [,nwords]

Displays nwords of hexadecimal and ASCII data.

DDECIMAL [/H], [name=] location [/type] [+offset] [,nwords]

Displays nwords of hexadecimal and decimal data.

DFLOAT [/H], [name=] location [/type] [+offset] [,nwords]

Displays nwords of hexadecimal and floating point data.

IDISPLAY, [name=] location [/type] [+offset] [,nwords]

Displays nwords of hexadecimal and ASCII data starting at the location indicated by the address specified by the location parameter (indirect addressing).

/H Indicates that the data to be displayed is half-word data.

name Name of a module, within the file, relative to which the location parameter is a reference; or 0, in which case the location is an absolute virtual address. An equals sign must immediately follow the name and precede the location, without intervening blanks, in the form name=location. Default name when DEBUG is first started is the main program (or the first module loaded, for non-FORTRAN-generated code); otherwise, the default name is that name last referenced; in the case that the last reference was of the form 0=location, the location is assumed to be an absolute address and an associated type of S or L is disallowed.

location A hexadecimal address, source line number, statement label, simple variable name, descriptor name, or array name, indicating location at which display is to originate or, for IDISPLAY, the location containing the address indicating the location at which display is to originate. When the offset parameter is present, the location parameter indicates a location relative to which display is to originate.

type One of the following characters defining the type of location designated:

S Statement label, simple variable name, descriptor name, or array name (FORTRAN programs only)

L Source line number (FORTRAN programs only)

X Hexadecimal bit address

W Hexadecimal word address

P Hexadecimal page address

Default type when DEBUG is first started is S; otherwise, the default type is that type last referenced.

offset Hexadecimal number, indicating an upward or downward offset, in words, from the location indicated by the location parameter. A plus sign or minus sign must immediately precede the number.

nwords Hexadecimal value designating the number of words or half-words to be displayed. Default value is #4; maximum allowed value is #10.

When the controllee file for DEBUG is a FORTRAN program that has been compiled without the S option, dynamic space fields, variables in common areas, and variables that are parameters can be displayed and altered using DEBUG directives. Variables in areas declared common can be displayed by referencing them in the module specified or last referenced. Referencing a descriptor associated with the currently allocated dynamic space fields for the breakpointed module and its higher level modules (modules that have led to the call to the breakpointed module and are linked to it through previous stack pointers) displays the contents of those fields. Variables that are parameters in the breakpointed module can be displayed by referencing them in the usual way after the prologue of the breakpointed module has been executed and the variables thereby set to their passed values; during the prologue, their values are indeterminate.

The following directives display virtual memory forward or backward from the last display location:

ROLL[,nwords] Displays area following last location.

BACK[,nwords] Displays area preceding last location.

nwords Hexadecimal value designating the number of words to be displayed starting from last location displayed. Default value is the number of words specified by the previous directive; maximum allowed value is #10.

Register Directives

The user can display and alter the contents of the user program registers by issuing one of the following:

<u>D</u> REG,hexreg[,nregisters]	Displays the contents of a register.
<u>E</u> REG,hexreg,hexdata	Allows user to enter hexadecimal data into a register.
<u>I</u> DREG,hexreg[,nwords]	Displays data found at the address that is given in specified register.
hexreg	Full-word hexadecimal register number which contains data to be displayed or into which data is to be entered.
nregisters	Specifies hexadecimal number of registers to be displayed, starting with hexreg. Default value is #4; maximum value allowed is #10.
hexdata	Hexadecimal half-word data to be entered into n consecutive registers starting with high-order half of hexreg. Values are right-justified with zero fill.
nwords	Hexadecimal value designating the number of words to be displayed. Default value is #4; maximum value allowed is #10.

Alter Memory Directives

The user can alter virtual memory by entering one of the following:

<u>H</u> EX,[name=] location [/type][+_offset],halfhex	Enters hexadecimal data.
<u>A</u> SCII,[name=] location [/type][+_offset], "ASCIIdata"	Enters an ASCII character string.
<u>D</u> ECIMAL [/H],[name=] location [/type][+_offset],decidata	Enters decimal data.
<u>F</u> LOAT [/H],[name=] location [/type][+_offset],fltpt	Enters floating point data.
/H	Indicates that the data to be displayed is half-word data.

name Name of a module, within the file, relative to which the location parameter is a reference; or 0, in which case the location is an absolute virtual address. An equals sign must immediately follow the name and precede the location, without intervening blanks, in the form name=location. Default name when DEBUG is first started is the main program (or the first module loaded, for non-FORTRAN-generated code); otherwise, the default name is that name last referenced. In the case that the last reference was of the form 0=location, the location is assumed to be an absolute address and an associated type of S or L is disallowed.

location A hexadecimal address, source line number, statement label, simple variable name, descriptor name, or array name, indicating location at which data is to be entered. When the offset parameter is present, the location parameter indicates a location relative to which the data is to be entered.

type One of the following characters defining the type of location designated:

- S Statement label, simple variable name, descriptor name, or array name (FORTRAN programs only)
- L Source line number (FORTRAN programs only)
- X Hexadecimal bit address
- W Hexadecimal word address
- P Hexadecimal page address

Default type when DEBUG is first started is S; otherwise, the default type is that type last referenced.

offset Hexadecimal number, indicating an upward or downward offset, in words, from the location indicated by the location parameter. A plus sign or minus sign must immediately precede the number.

halfhex Half-word hexadecimal data values to be entered into consecutive half-word memory locations starting at location specified. Values are right-justified with zero fill.

ASCIIdata String of ASCII data to be entered into consecutive character locations starting at the position given by location parameter. The ASCII data string must be enclosed in quotation marks.

decidata Full- or half-word decimal data to be entered into consecutive full- or half-word memory locations beginning at the location specified.

fltpt Floating point data to be entered into consecutive half- or full-word memory locations, depending on data type parameters, starting at location specified. E or F format can be used.

Program Control Directives

The user can set and remove breakpoints to start and stop program execution, to dump portions of virtual memory to an output file, or to find the status of DEBUG directives issued earlier by issuing one of the following:

BKPT,[name=] location [/type] [+offset]

Defines a breakpoint. User program execution stops before the instruction located at the breakpoint address is executed.

BKPTR,[name=] location [/type] [+offset]

Removes a breakpoint. If no parameters are given, all breakpoints in the program are removed.

EXECUTE,[name=] location [/type] [+offset]

Causes DEBUG to start executing the user program at the location specified. If no parameters are given on the EXECUTE directive, DEBUG starts at the transfer address. Only one EXECUTE directive can be given per DEBUG run and it must appear before any CONTINUE directive.

CONTINUE

Causes user program execution to be continued from the last breakpoint encountered. CONTINUE can appear only after EXECUTE or another CONTINUE directive.

STEP,[ninstr]

Causes user program execution to continue for ninstr number of instructions from the last breakpoint encountered.

END

Terminates both the user program and DEBUG.

SNAP,[name=] location [/type] [+offset] [,nwords]

Dumps nwords number of words of virtual memory, starting from location, to an output file. Output data is in hexadecimal and ASCII.

SNAP,hexreg,R [,nwords]

Dumps the contents of nwords number of registers, starting with register numbered hexreg. Output is in hexadecimal and ASCII.

name Name of a module, within the file, relative to which the location parameter is a reference; or 0, in which case the location is an absolute virtual address. An equals sign must immediately follow the name and precede the location, without intervening blanks, in the form name=location. Default name when DEBUG is first started is the main program (or the first module loaded, for non-FORTRAN-generated code); otherwise, the default name is that name last referenced; in the case that the last reference was of the form 0=location, the location is assumed to be an absolute address and an associated type of S or L is disallowed.

location A hexadecimal address, source line number, statement label, simple variable name, descriptor name, or array name, indicating location at which data is to be entered. When the offset parameter is present, the location parameter indicates a location relative to which the data is to be entered.

type One of the following characters defining the type of location designated:

S Statement label, simple variable name, descriptor name, or array name (FORTRAN programs only)

L Source line number (FORTRAN programs only)

X Hexadecimal bit address

W Hexadecimal word address

P Hexadecimal page address

Default type when DEBUG is first started is S; otherwise, the default type is that type last referenced.

offset Hexadecimal number, indicating an upward or downward offset, in words, from the location indicated by the location parameter. A plus sign or minus sign must immediately precede the number.

ninstr Hexadecimal number specifying a number of instructions. Default value is #1; the maximum value allowed is #10.

nwords Hexadecimal number specifying a number of words or registers. Default value is #4.

STAT

Produces a listing of the breakpoints set, the last DEBUG and BKPT directive issued, the last routine name or common block referenced, the last type referenced, the next execution address in the user program, and displays the last module referenced.

LOOK

The LOOK utility can be used to examine statically any mass storage file to which the user has access. It cannot be used during program execution.

After LOOK execution has been initiated with the LOOK control statement, the utility responds to directions received from directives. The modifications made through LOOK directives persist for the life of the file modified. LOOK remains in execution until an END is received.

LOOK CONTROL STATEMENT

The control statement that initiates execution of LOOK is shown in figure 6-4. The parameter fname must always be the first parameter, but the order of the I= and L= parameters can be reversed.

LOOK DIRECTIVES

All numbers in all LOOK directives, unless otherwise specified, must be hexadecimal. Directives can be concatenated by slashes. For example, V/W/HEX,1000,10 is valid, and it indicates three directives that LOOK would process consecutively, just as if they had been issued separately in the same order.

When the parameters for a given directive are meaningless, missing, or illegal, the directive is ignored and an error message is issued.

The response format for the directive is:

COMMAND=command keyword
output

Examples of LOOK directives are shown in figure 6-5.

LOOK,fname,I=iname,L=onname/olen/disp.

fname	Name of the existing mass storage file being modified or examined by LOOK.
I=iname	Optional. Name of an existing ASCII file containing directives for LOOK. Default is I=INPUT.
L=onname/olen/disp	Optional. For batch mode, describes the file to which all LOOK output is written. For interactive mode, describes the file to which output is written when the PRINT directive is issued:
oname	Name of the file. Default is OUTPUT.
olen	Integral length of oname in small pages, in decimal; must be greater than 0 and less than 1001. Default is 128 small pages.
disp	Disposition of oname, currently PR only. Default is PR.
If oname exists already, it is destroyed before being created.	

Figure 6-4. LOOK Control Statement Format

PRINT/PAGE/HEX,0,8	PRINT indicates that all successive output is to be written to the file OUTPUT. PAGE indicates that all addresses entered subsequently are page addresses. HEX,0,8 causes a dump of eight full words from the beginning of the first page in the file.
HEX,,8	Display eight words at the beginning of the first page in the file, assuming the previous PAGE directive. The three columns displayed indicate the word address, hexadecimal contents, and contents in ASCII.
SEARCH,'/',,3	Search the file for the third appearance of the character /, beginning at the start of the file. The address at which the specified occurrence is found is reported on the output file.
BIT/EASCII,C0,'1111'/HEX,C0,1	BIT indicates that all addresses entered subsequently are bit addresses. EASCII places the ASCII character string 1111 at the halfword address C0. The next directive displays the word just entered in hexadecimal and ASCII.

Figure 6-5. Sample LOOK Directives

In the individual descriptions given listed in the following, optional parameters can be omitted and a default value will be assigned. If an embedded parameter is to be omitted, commas must be used to maintain positional integrity; for example, HEX,,2 defaults the beginning address to 0. If the last one or more parameters are to be omitted, the command must end with the last specified parameter. For example, HEX defaults address to 0 and length to 1; HEX,100 defaults length to 1.

LOOK directives are listed in the following. After this list, the directives are described in a logical grouping rather than in alphabetical order.

<u>BACK</u>	Display file portion that immediately precedes portion last displayed.
<u>BASE</u>	All addresses in other LOOK directives are to be offset by a specified amount.
<u>BIT</u>	All addresses in other LOOK directives are to be interpreted as bit addresses.
<u>DEC</u>	Dump or display file in full-word decimal format.
<u>DISPLAY</u>	All output is to go to the terminal.
<u>EASCII</u>	Enter ASCII character string in file at specified location.
<u>EDEC</u>	Enter full-word decimal data items into file at specified location.
<u>EFLOAT</u>	Enter full-word floating point data items into file at specified location.
<u>EHDEC</u>	Enter half-word decimal data items into file at specified location.
<u>EHEX</u>	Enter half-word hexadecimal data items into file at specified location.
<u>EHFLOAT</u>	Enter half-word floating point data items into file at specified location.
<u>END</u>	End LOOK.
<u>FLOAT</u>	Dump or display file in full-word floating point format.
<u>HDEC</u>	Dump or display file in hexadecimal and half-word decimal format.
<u>HEX</u>	Dump or display file in hexadecimal and ASCII format.
<u>HFLOAT</u>	Dump or display file in hexadecimal and half-word floating point format.
<u>IDEC</u>	Dump or display file in full-word decimal format (indirect addressing of file).
<u>IFLOAT</u>	Dump or display file in hexadecimal and half-word floating point format (indirect addressing of file).

<u>IHDEC</u>	Dump or display file in hexadecimal and half-word decimal format (indirect addressing of file).
<u>IHEX</u>	Dump or display file in hexadecimal and ASCII format (indirect addressing of file).
<u>IHFLOAT</u>	Dump or display file in hexadecimal and half-word floating point format (indirect addressing of file).
<u>PAGE</u>	All addresses in other LOOK directives are to be interpreted as page addresses.
<u>PATTERN</u>	Enter pattern into portion of file.
<u>PRINT</u>	All output from directives is to go to the output file.
<u>ROLL</u>	Display file portion that immediately follows portion last displayed.
<u>SEARCH</u>	Search file for appearance n of a specified character string.
<u>SEQ</u>	All addresses in other LOOK directives are to be interpreted as sequential from this point on.
<u>VIRTUAL</u>	All addresses in other LOOK directives are to be interpreted as being virtual.
<u>WORD</u>	All addresses in other LOOK directives are to be interpreted as word addresses.

SEARCH Directive

LOOK searches a file for an occurrence of a specified string of characters when the SEARCH directive is issued:

SEARCH, 'string', [addr], [n]

string A string of ASCII characters. The string must be enclosed in single quotes.

addr Position in file where search for the character string is to begin, designated by a byte address. If addr does not lie on a byte boundary, it is truncated to the nearest byte boundary. Default is 0.

n Integer constant greater than zero, indicating the occurrence of the character string that is to be selected. Default is 1.

Beginning at addr, the file is searched for n occurrences of string, and the address of the last is returned. If LOOK is in the virtual mode, the search is through the file virtually. If LOOK is in the sequential mode, the search is through the file sequentially.

If no occurrences of the character string are found, the message

CHARACTER STRING NOT FOUND

is issued. If m, but fewer than n, occurrences are found when the end of the file is reached, the message

ONLY m OCCURRENCES WERE FOUND

is issued. If the search is successful, a message

CHARACTER STRING FOUND AT address

is issued, where address is the bit address of the first character of the selected string.

Disposition of Directive Output

The interactive user can select whether output is to be displayed at the terminal or dumped onto the output file that was specified in the LOOK control statement. Initial mode is DISPLAY. The directives that control the disposition of output are:

<u>PRINT</u>	This indicates that from the time of this directive, or until a DISPLAY or END directive is entered, all output is to be written to the output file.
<u>DISPLAY</u>	This indicates that from the time of this directive, or until a PRINT or END directive is entered, all output is to be sent to the terminal.
<u>END</u>	End LOOK.

Display and Dump Directives.

The following directives cause dump or display of a specified number of words of the file starting at a specified location in the file. For interactive users in display mode, the display is one word or half-word per line. On the output file, four words are placed in one line, with duplicate lines being suppressed. The directives are:

<u>DEC</u> ,[addr],[len]	Displays/dumps file portion as hexadecimal and full-word decimal data.
<u>FLOAT</u> ,[addr],[len]	Displays/dumps file portion as hexadecimal and full-word floating point data.
<u>HDEC</u> ,[haddr],[len]	Displays/dumps file portion as hexadecimal and half-word decimal data.
<u>HEX</u> ,[haddr],[len]	Displays/dumps file portion as hexadecimal and ASCII data.
<u>HFLOAT</u> ,[haddr],[len]	Displays/dumps file portion as hexadecimal and half-word floating point data.

<u>addr</u>	Position in file where display/dump is to begin, designated by a word address. If addr is not on a word boundary, it is truncated to the nearest word boundary. Default is 0.
<u>haddr</u>	Position in file where display/dump is to begin, designated by a half-word address. If haddr is not on a half-word boundary, it is truncated to the nearest half-word boundary. Default is 0.
<u>len</u>	The number of words displayed/dumped. If len is not specified, it is taken to be 1.

Corresponding to each of the preceding directives is another LOOK directive. This directive performs the identical operation, except that the address parameter specified must be the indirect, rather than the direct, address of the position where display or dump is to begin. The directives are:

<u>IDEC</u> ,[addr],[len]	Displays/dumps file portion as hexadecimal and full-word decimal data.
<u>IFLOAT</u> ,[addr],[len]	Displays/dumps file portion as hexadecimal and full-word floating point data.
<u>IHDEC</u> ,[addrh],[len]	Displays/dumps file portion as hexadecimal and half-word decimal data.
<u>IHEX</u> ,[addrh],[len]	Displays/dumps file portion as hexadecimal and ASCII data.
<u>IHFLOAT</u> ,[addrh],[len]	Displays/dumps file portion as hexadecimal and half-word floating point data.
<u>addr</u>	Address of word containing position in file where display/dump is to begin; if addr is not a full-word address, the value of addr is interpreted to be the first word boundary preceding addr. The low 48 bits of the word at addr is the file position where the display or dump begins; if the 48 bits do not constitute a word address, the display/dump begins on the first word boundary preceding the address. Default is 0.
<u>haddr</u>	Address of word containing position in file where display/dump is to begin; if haddr is not a full-word address, the value of haddr is interpreted to be the first word boundary preceding haddr. The low 48 bits of the word at haddr is the file position where the display or dump begins; if the 48 bits do not constitute a half-word address, the display/dump begins on the first halfword boundary preceding the address. Default is 0.
<u>len</u>	The number of words displayed/dumped. If len is not specified, it is taken from the top 16 bits of the word at addr or haddr. Default is 1.

Additional data can be displayed or dumped with either of the following:

ROLL Displays or dumps the file portion that immediately follows the portion last displayed or dumped. The number of words and format is the same as that of the previous display/dump.

BACK Displays or dumps the file portion that immediately precedes the portion last displayed or dumped. The number of words and format is the same as that of the previous display/dump.

Directives for Entering Values

LOOK offers seven directives for entering values into the file. The directives cause values to be placed in the file beginning at a particular location. The operation performed is not an insertion; that is, does not cause expansion of the size of the file, but is, instead, a replacement of the current value with another.

Any file on which these operations are performed must have write access. The directives are:

PATTERN,haddr,laddr,data₀[data₁...data_n]

Enters half-word data pattern into the file between haddr and laddr inclusive.

EASCII,haddr,'string'

Enters character string starting at haddr.

EDEC,haddr,data₀[data₁...data_n]

Enters full-word decimal data items starting at haddr.

EFLOAT,addr,data₀[data₁...data_n]

Enters full-word floating point data items starting at addr.

EHDEC,haddr,data₀[data₁...data_n]

Enters half-word decimal data items starting at haddr.

EHEX,haddr,data₀[data₁...data_n]

Enters half-word hexadecimal data starting at haddr.

EHFLOAT,haddr,data₀[data₁...data_n]

Enters half-word floating point data items starting at haddr.

haddr Position in file where entry is to begin, designated by a hexadecimal half-word address; haddr must be on a half-word boundary.

laddr Position in file where the last half-word entered is to be placed, designated by a hexadecimal half-word address; laddr must be on a half-word boundary.

data_i The data to be entered; either a half-word or a full-word of data, depending on the directive. Depending on the directive, data_i is a hexadecimal, decimal, or floating point number that can be represented in a half-word or full-word.

addr Position in file where entry is to begin, designated by a hexadecimal full-word address; addr must be on a full-word boundary.

string A string of ASCII characters. The string must be enclosed in single quotes. If the number of characters in the string is not a multiple of 4, the last half-word is blank filled on the right.

Declaration of Directive Address Type

The user can indicate whether addresses in LOOK directives specify a quantity of bits, words, or pages by entering one of the following:

WORD Specifies that all addresses in LOOK directives are to be interpreted as word addresses.

PAGE Specifies that all addresses in subsequent LOOK directives are to be interpreted as small page addresses.

BIT Specifies that all addresses in subsequent LOOK directives are to be interpreted as bit addresses.

The user can indicate that all addresses in LOOK directives are to be offset by entering:

BASE,offset

offset Offset to be added to all addresses. The offset is in bits, words, or page, depending on the address mode established by BIT, WORD, or PAGE directives.

Initially, BASE is 0.

The user can indicate whether the virtual file being manipulated is to be accessed as a virtual or physical file by entering one of the directives:

SEQ Declares that from this point on, or until VIRTUAL or END is entered, all addresses referred to in other directives are sequential addresses. This allows access to the one or two minus pages of a virtual file, which are not virtually addressable.

VIRTUAL Declares that from this point on, or until SEQ or END is entered, all addresses referred to in other LOOK directives are virtual addresses.

For physical files, these directives are meaningless and if entered are ignored; physical files are always in SEQ mode. At the beginning of a LOOK run, mode is VIRTUAL (unless the file is physical).

These specifications hold until another of these directives is entered or LOOK ends. The initial address mode is BIT.

DUMP

When a fatal error occurs during execution of a program, a dump of information extracted mostly from the drop file for the executing program can be placed in the output file for the program. This dump is performed automatically by the batch processor. The standard items in the dump are the following, in the indicated order:

1. Program address.
2. Last executed instruction in hexadecimal.
3. System error code of the fatal error condition and the address indicating where the error occurred (also returned in word 139 of the first minus page). Codes are defined in volume 2.
4. Subroutine traceback. If the error occurred in a subroutine, the address of each CALL statement that led to that subroutine is listed.
5. List of the open (active) files at the time the error occurred along with each file's virtual page address and its length in pages.
6. Alpha and Beta words if the word preceding the program address contains an exit force instruction.

7. Contents of the first minus page in hexadecimal and ASCII. Duplicate lines are suppressed.
8. Contents of the second minus page in hexadecimal and ASCII.
9. Contents of memory in hexadecimal and ASCII, from 50 words preceding the program address to 50 words following the address.
10. If the error occurred in a subroutine, the register save area for each caller is dumped; then, the contents of memory from 50 words preceding to 50 words following the caller's address in the subroutine traceback list is dumped. Dumps are output in reverse sequence of the CALL occurrences that led to the failing subroutine. Subroutines are always dumped, regardless of whether they are system-supplied or user-supplied.

On some fatal error conditions, no dump is made automatically. If no dump is made, the user has the option of requesting the dump if the drop file for the program has persisted. The FILES control statement can be issued to determine if the drop file has been retained automatically as one of the user's private files. If the file has been retained, the FILES control statement can be issued to determine the drop file name for use in the DUMP control statement.

The format of the DUMP control statement is given in figure 6-6. It can be issued either interactively or from within a batch control sequence.

DUMP,dropfile.

dropfile Name of program's drop file.

Figure 6-6. DUMP Control Statement Format

The checkpoint/restart capability allows a task to be restarted from some intermediate point in the event of abnormal task termination. The facility provides a means of conserving machine time from a second execution of a long task. Through a call to CHKPNT, a programmer captures the status of an executing task and any of its controllees; if necessary, the chain of tasks can be restarted from the checkpoint information captured by the call.

When checkpoint is called, the system makes copies of the drop files associated with the checkpointed task and all of its controllees. Checkpoint information includes a list of controllees to be restarted. Checkpoint file names are formed by the system modifying the file name in the checkpoint call with an ending digit 1 through 5; this indicates the controllee level relative to the task with the checkpoint call.

Checkpoint restores system message control for the checkpointing task and tasks at lower levels in its controllee chain. Any message outstanding when a checkpoint occurs is not preserved, however. For a controllee with a message bypass pointing to a controller of higher level than the checkpointing task, the bypass is set by restart to point to the checkpointing task. Refer to the Message Control system message in volume 2.

CHKPNT - CHECKPOINT CALL

A checkpoint is taken by the system in response to a call to CHKPNT. The programmer is responsible for ensuring that the checkpoint occurs at a logical breaking point during task execution.

The CHKPNT subroutine format is shown in figure 7-1.

All files created by CHKPNT are local files. The local files can be made permanent with DEFINE.

A task can call CHKPNT more than once with a given file name for the lfn variable. At each such call to CHKPNT, a copy of the current state of the drop file is created. When CHKPNT terminates successfully, this copy is assigned the name lfn, and the previous checkpoint file is destroyed. If program execution is aborted before checkpoint terminates, the previously existing file is retained as the checkpoint file.

The intermediate checkpoint file name at the calling controllee level is Q5CKP1; if lower level controllees exist, their intermediate checkpoint file names are Q5CKP2 through Q5CKP5.

CALL CHKPNT (lfn, rest, err, erlfn)

lfn	Name of file to which the checkpoint information is to be written. Must be 1 through 8 letters or digits beginning with a letter expressed as a Hollerith constant or a variable left-justified and blank filled.
rst	Variable to which the system returns a response after checkpoint and restart: 0 Checkpoint executed. 1 Restart executed.
err	Variable to which the system returns a code for any error found in either checkpoint or restart.
erlfn	Variable to which the system returns the name, left-justified and blank filled, of any file in which errors were encountered during checkpoint or restart. The particular error that causes this variable to be set is documented in the err variable.

Figure 7-1. CHKPNT Subroutine Format

Checkpoint error responses that can be returned in the err variable established by the call to CHKPNT are:

- 0 No error.
- 1 Checkpoint file name specified in erlfn duplicates an existing local or attached permanent file name.
- 2 Input/output connector error.
- 3 Mass storage or system table space not available.
- 4 Invalid file name supplied by program.
- > 4 System error with return word formatted as in Q5STATUS call.
- 1 Error in bound implicit map entry for file OUTPUT.

Restart error responses that can be returned in the err variable established by the call to CHKPNT are:

- 0 No error.
- 10 Controllee cannot be initialized because system tables are full.
- 11 More than five levels of controllees.
- 13 Controllee file specified in erlfn cannot be found.
- 14 Insufficient time to run controllee specified in erlfn.
- 15 System error; restart failed.
- 17 Controllee file specified in erlfn is not executable.
- 18 Mass storage error for file specified in erlfn.
- 19 Error in controllee file or drop file input/output connector specified in erlfn.
- >19 System error with return word formatted as in Q5STATUS call.

RESTART

A checkpointed task is restarted through a control statement that names the file containing checkpoint information. The system restarts all controllees in response to the single call. Once the controllee chain is reestablished, the system returns control to the checkpointed program at the statement immediately following the checkpoint call.

Restart verifies that all files open at checkpoint time still exist and still occupy the same file space. The file OUTPUT is recreated if necessary. Any tape files attached at checkpoint time are not reattached by restart, however. The restarted task is responsible for reestablishing connection with any tape and for repositioning as required.

A task containing several calls to checkpoint with a given file name might run into conflict during the restart phase. If the restart file is itself named lfn, it cannot call checkpoint with the same lfn. For this case, it is recommended that a switch to another name of the restart file be done before commencing its execution. If this is not done, however, the checkpoint files are still created using the intermediate checkpoint file names.

Sections 8 and 9 describe the System Interface Language (SIL). SIL is a set of subroutines callable by user programs. The user program can be written in FORTRAN, IMPL, or the META Assembler language. Each subroutine formats and issues a system message. (System messages are described in volume 2.) The routines described in section 9 perform functions related to I/O. The routines described in this section perform non-I/O functions enabling a task to exchange information with the operating system.

OVERVIEW

The following is a list of the routines described in this section grouped according to a shared function.

Inform the system of the task's requirements.

- Q5ADVISE Informs the system of the task's virtual space requirements.
- Q5DESBIF Informs the system that the task should not be rerun if the system fails.
- Q5DMPACT Dumps the cumulative accounting file and terminates its use.
- Q5RECALL Suspends task execution.
- Q5RUNBIF Informs the system that the task should be rerun if the system fails.
- Q5SETLP Changes the current large page limit.
- Q5VRACC Changes the task's accounting rate.

Enable the task to communicate with the system operator.

- Q5GETMOP Gets a message sent by the system operator.
- Q5SNDMOP Sends a message to the system operator.

Determine task processing if the task encounters an error.

- Q5DISATI Disables abnormal termination control.
- Q5ENATI Enables abnormal termination control.
- Q5RFI Returns control from an interrupt routine.

Determine message interrupt processing for the task.

- Q5DISAMI Disables message interrupt processing.
- Q5ENAMI Enables message interrupt processing.

- Q5RFI Returns control from an interrupt routine.

Initialize and terminate a controllee chain.

- Q5INIT Initializes or initializes and starts a controllee.
- Q5INITCH Initializes a controllee chain.
- Q5SNDSTR Starts controllee execution.
- Q5TERM Terminates a task and its controllee chain.
- Q5TERMCE Disconnects a controllee.

Control message flow within a controllee chain.

- Q5GETMCE Gets message from controllee.
- Q5GETMCR Gets message from controller.
- Q5MSGCTR Redirects messages sent to a task.
- Q5SNDMCE Sends message to controllee.
- Q5SNDMCR Sends message to controller.
- Q5SNDMDMF Sends message to job dayfile.
- Q5SNDMJC Sends message to job controller.

Get information about the controllee chain.

- Q5GETCTS Gets the controllee's termination status.
- Q5LSTCH Gets information about each task in the controllee chain.

Get information about the system and the task.

- Q5CPUTIM Gets the CPU time the task has used.
- Q5DCDDST Gets and decodes information from the Disk Status Table.
- Q5DCDMSC Gets and decodes information from the Miscellaneous Table.
- Q5DCDPFI Decodes permanent file indices.
- Q5DCDPLB Decodes a pack label.
- Q5GETACT Gets the system resources the task has used.
- Q5GETLP Gets the task's large page limits.

Q5GETTL	Gets the task's time limit.
Q5GETTN	Gets the task's characteristics.
Q5GETUID	Gets the user number under which the task is running.
Q5TIME	Gets the system time and date.

Get permanent file indices to be decoded by Q5DCDPFI.

Q5GETPFI	Gets the label and permanent file indices from a pack.
Q5LFIPOL	Gets the permanent file indices for pool files.
Q5LFIPRI	Gets the permanent file indices for private files.
Q5LFIPUB	Gets the permanent file indices for public files.

Copy system tables to user-defined arrays.

Q5GETIIP	Copies the interrupted task's invisible package.
Q5GETIRF	Copies the task's register file.
Q5GETMPG	Copies the interrupted task's minus page information.
Q5LSTBUT	Copies the Bank Update Table.
Q5LSTSTB	Copies the Statistics Buffer.
Q5LSTTCB	Copies the Timecard Buffer.

SIL ERROR PROCESSING

The user can specify three parameters that return information on the last error (if any) encountered during call processing. The three parameters are:

- 'STATUS=',stat
- 'ERRMSG=',msg
- 'ERRLEN=',len

SIL returns the status code of the last error encountered in the variable stat (if specified).

The status code categories are as follows.

<u>Range</u>	<u>Meaning</u>
0	No errors.
1-199	User parameter specification error. The code number is the ordinal of the parameter within the calling sequence.
200-249	Internal error. Consult a systems analyst.

<u>Range</u>	<u>Meaning</u>
250-9999	Error detected by SIL or the operating system.
10000-11000	Error defined by the installation.

Each status code is listed with its associated error message in appendix B.

SIL returns an 80-byte error message in the variable msg (if specified). It always assumes msg is 80 bytes long. The possible error messages are listed in appendix B. If the ERRLEN= parameter is specified, SIL also returns the length of the actual error message (excluding trailing blanks).

Each error has a severity level, either fatal or warning. Warning errors return control to the caller. Fatal errors also return control to the caller if the STATUS= parameter is specified on the call. If the STATUS= parameter is omitted from the call, the task is aborted, returning control to the controller with a termination value of 8 (refer to Error Processing in section 3).

Error message routing depends on whether the ERRMSG= parameter is specified and on whether the task is to be aborted as a result of the error. The possible actions are summarized as follows.

	<u>ERRMSG=,msg specified</u>	<u>ERRMSG=,msg omitted</u>
Task to be aborted	Message sent to controller and to msg variable.	Message sent to controller.
Task not to be aborted	Message sent to msg variable.	Message sent to controller.

Messages sent to the task's controller usually appear in the dayfile of a batch job or at the terminal of an interactive task.

SIL CALL FORMAT

Calls to SIL routines have the following general format.

CALL Q5xxxxxx(p₁,p₂,...,p_n)

Q5xxxxxx is the name of an SIL routine. SIL parameters, p_i, have two formats, paired and stand-alone.

Paired parameters have two parts, a keyword and an option, separated by a comma. The keyword of a paired parameter always ends with an = character. The user can specify a keyword as a literal or as a variable name containing the keyword. Depending on the parameter, the option may be character data such as an identifier or mnemonic, numeric data such as a buffer or file length, an array address specified by a subscripted variable name, the name of a descriptor, or the name of an external subroutine.

The methods of specifying FORTRAN data formats are described in the CYBER 200 FORTRAN Reference Manual. Methods of specifying IMPL data formats are described in the CYBER 200 IMPLERS.

For example, the following call sends a message to the controller. The message is 'READING TAPE' and is 12 bytes in length. The request status is returned in a variable called STATUS.

```
CALL Q5SNDMCR('MSG=', 'READING TAPE',
              'LEN=', 12, 'STATUS=', STATUS)
```

Stand-alone parameters consist of a keyword only. The keyword does not end with an = character. A stand-alone parameter indicates a logical value (yes or no, on or off) by its presence or omission. The user can specify the parameter as a literal ('WAIT') or as the name of the variable containing the parameter (refer to the example in the description of no-operation keywords).

The FORTRAN and IMPL compilers convert the call formats shown in this section to the appropriate calling conventions as described in appendix D of volume 2. The descriptor of each character parameter (calling or return) must contain a valid length portion specifying the length of the parameter in bytes. SIL considers a length portion of zero as equivalent to a length portion of eight. If the length specified (or assumed) is longer than the actual literal, the literal must be left-justified and the remaining characters blanks.

SIL parameters are either required or optional. Required parameters are specified in the individual call formats given later in this section. If the user omits a required parameter in a call, SIL returns a fatal error. Certain calls require one parameter specified from a required set. If no parameter is specified from the set, the error message specifies the set number. If the user omits an optional parameter, SIL uses the default value.

Certain calls forbid specification of more than one parameter of a mutually exclusive set of parameters. If more than one parameter is specified from the set, the error message indicates the second parameter specified from the set is illegal.

When the user specifies a variable containing character code data, the contents of the variable must be left-justified and blank-filled to the length specified (or eight bytes). Binary numeric data (such as buffer and file lengths) must be right-justified and zero-filled.

Keyword and byte (8-bit bytes) parameters begin on byte boundaries unless otherwise stated in the descriptions. Binary numeric data must begin on word boundaries.

NO OPERATION KEYWORDS

SIL recognizes two special no operation keywords, 'NOP=' for paired parameters and 'NOP' for stand-alone parameters.

The following example illustrates use of the NOP keyword. The following FORTRAN code tests a condition and then either terminates a task abnormally, resulting in an error being issued, or normally, with no error issued.

```
EC = 'NOP'
IF (ERROR) EC = 'ABORT'
CALL Q5TERM (EC)
```

SIL NON-I/O CALLS

This section contains a figure for each SIL routine. The figure contains a call format specifying the required parameters followed by parameter descriptions. The parameter descriptions are divided between calling parameters and return parameters. A calling parameter specifies a value used by the SIL routine. A return parameter specifies the name of the variable in which SIL returns a value.

Parameter keywords are listed as FORTRAN literals in the call formats.

Q5ADVISE-ADVISE SYSTEM OF VIRTUAL SPACE REQUIREMENTS

The Q5ADVISE subroutine (refer to figure 8-1) informs the operating system of the task's current virtual space needs. The system uses this information to minimize task paging requirements. It keeps paged in the virtual space the task indicates it needs and pages out the virtual space the task indicates it no longer needs.

Q5ADVISE recognizes the following two methods of specifying a virtual region.

- By a descriptor word containing the address and length of the virtual region. (Descriptor declaration and initialization is described in the CYBER 200 FORTRAN Reference Manual.)
- By the name of the first array element in the virtual region. The user can specify the length of the region; if he does not, Q5ADVISE assumes the default value.

Q5ADVISE uses the Advise system message.

Examples:

The following FORTRAN source lines illustrate the two methods of specifying a virtual region. In each case, the Q5ADVISE call advises the system that it should keep ARRAY1 paged in, but it can page out ARRAY2.

Virtual region specified by address:

```
DIMENSION ARRAY1(1024), ARRAY2(1024)
CALL Q5ADVISE('INADDR=', ARRAY1(1),
              + 'INLEN=', 1024, 'OUTADDR=', ARRAY2(1),
              + 'OUTLEN=', 1024)
      .
      .
      .
```

Virtual region specified by descriptor:

```
DIMENSION ARRAY1(1024), ARRAY2(1024)
DESCRIPTOR ADRIN, ADROUT
DATA ADRIN, ADROUT/ARRAY1(1;1024),
+ ARRAY2(1;1024)/
CALL Q5ADVISE('INDESC=', ADRIN,
              + 'OUTDESC=', ADROUT)
      .
      .
      .
```

Call Format

CALL Q5ADVISE($\left\{ \begin{array}{l} \text{'INADDR='}, \text{addr} \\ \text{'INDESC='}, \text{desc} \\ \text{'OUTADDR='}, \text{addr} \\ \text{'OUTDESC='}, \text{desc} \end{array} \right\}$, optional parameters)

Calling Parameters

'INADDR=',addr Starting virtual bit address of memory the program needs. If INDESC= is specified, the user must omit INADDR=.

'INDESC=',desc Descriptor containing both the length and address of the memory the program needs. The upper 16 bits of the word contain the memory length; the lower 48 bits contain the starting virtual bit address. If INADDR= is specified, the user must omit INDESC=.

'INLEN=',len Integer number of memory words the program needs. If INADDR is specified but INLEN= is omitted, SIL assumes the task needs the 512 words starting at the address specified by INADDR=. If INLEN= and INDESC= are specified, SIL uses the length specified by the INLEN= parameter.

'OUTADDR=',addr Starting virtual bit address of the memory the program no longer needs. If OUTDESC= is specified, the user must omit OUTADDR=.

'OUTDESC=',desc Descriptor containing both the length and address of the memory the program no longer needs. The upper 16 bits of the word contain the memory length; the lower 48 bits contain the starting virtual bit address. If OUTADDR= is specified, the user must omit OUTDESC=.

'OUTLEN=',len Integer number of memory words the program no longer needs. If OUTADDR= is specified but OUTLEN= is omitted, SIL assumes the task needs the 512 words starting at the address specified by OUTADDR=. If OUTLEN= and OUTDESC= are specified, SIL uses the length specified by the OUTLEN= parameter.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299, 450 through 464.

Figure 8-1. Q5ADVISE Call Format

Q5CPUTIM-GET CPU TIME

The Q5CPUTIM subroutine (refer to figure 8-2) gets the CPU time (in microseconds) that the task has used. A task can issue this call ten times (unless the site changes the installation parameter setting the limit).

Q5CPUTIM uses the Miscellaneous system message.

Q5DCDDST-DECODE DISK STATUS TABLE

The Q5DCDDST subroutine (refer to figure 8-3) copies the Disk Status Table into a buffer it defines and retrieves information from the table copy.

SIL returns the information specified by the return parameters on the call. The user must specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=). All return information except the pack name is in binary format. The pack name is returned in character format.

Q5DCDMS-DECODE MISCELLANEOUS TABLE

The Q5DCDMS subroutine (refer to figure 8-4) copies the Miscellaneous Table into a buffer it defines and then retrieves information from the table copy.

SIL returns the information specified by the return parameters on the call. The user must specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=). All information is returned in binary, rather than character, format.

Call Format

CALL Q5CPUTIM(*TIME='time,optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299, 375.

'TIME=',time Task CPU execution time in microseconds (integer). The variable must be a full word on a word boundary. It is a required parameter.

Figure 8-2. Q5CPUTIM Call Format

Call Format

CALL Q5DCDDST(parameters)

NOTE

The user must specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

'ENTRY=',n Entry number (1 through 16) from which information is retrieved. If ENTRY= is omitted, the entry following the entry specified in the last Q5DCDDST call is used. If this call is the first Q5DCDDST call in the task, the first entry in the table is used.

Return Parameters

'DSTACT=',act Number of currently active files residing on this pack. (An active file is a file with an entry in FILEI.)

'DSTLOAD=',ver Version flag.

'DSTLU=',unit Physical unit number.

'DSTPFI=',res PFI residence flag.

0 PFI is not on this pack.
1 PFI is on this pack.

'DSTPFIL=',len Original length of PFI.

'DSTPFIO=',addr Page address of beginning of PFI.

'DSTPKLN=',len Pack length in blocks.

'DSTPRI=',flg Public pack flag.

0 Public pack.
1 Not a public pack.

'DSTTYP=',ns Nonstandard pack label field.

'DSTUP=',up Flag indicating if pack is logically up.

0 Pack is not up.
1 Pack is up.

'DSTZIP=',zip Zip code of station through which pack is accessed.

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array in which SIL returns an error message.

'NPACK=',n Number of entries in the Disk Status Table.

'PN=',pn Pack name in ASCII. The variable must begin on a word boundary.

'STATUS=',stat Status code. Possible values: 0 through 299.

Figure 8-3. Q5DCDDST Call Format

Call Format

CALL Q5DCDMSC(parameters)

NOTE

The user must specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

'FELEN=',n Maximum number of front-end processors (currently 16). This parameter is required if the FECODES= or FEZIPS= parameter is specified.

Return Parameters

'A=',alt Alternator currently running.

'AA=',alt Alternator for which the virtual system is running.

'ATIME=',time Time in the format (hh.mm.ss ASCII characters). The variable must begin on a word boundary.

'CALDLAY=',flg Write history delay flag.

'CEUSRNO=',un User number (in ASCII) that can run on-line diagnostics. The variable must begin on a word boundary.

'DATE=',date Date in ASCII characters. The variable must begin on a word boundary.

'DBLOCK=',db Descriptor block number for whom all pages are locked.

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array in which SIL returns an error message.

'FECODES=',fe Array containing the front-end processor types. The array length is returned in the FELEN= variable. Possible binary values returned:

0000	None.
X0X1	Service station.
XX11	Unit record station.
X1X1	Access station.
1XXX	Link station.

'FEZIPS=',zip Array containing the zip codes of the front-end processors. The zip codes are hexadecimal values with zero fill. The array length is returned in the FELEN= parameter.

'HILGBLK=',n Number of large pages minus one.

'HISTBET=',b Beta word for history write.

'HISTEXT=',flg Record history flag.

'HISTOFF=',off History pointer offset.

'HISTSAV=',adr Starting sector address of the history file.

'HISTSIZ=',len Length of history file.

'HISTWRT=',flg Outstanding write history flag.

'HISTZIP=',zip Zipcode of the station containing the history file.

Figure 8-4. Q5DCDMSC Call Format (Sheet 1 of 3)

'IQMN=',iqm	User number of the input queue manager (integer).						
'LASTALT=',alt	Standby alternator.						
'LGPG=',n	Array containing the number of small pages in each large page. Hexadecimal values with zero fill are returned. The array length is returned in the NLGPG= variable.						
'LPAV=',av	Array containing availability indicators for each large page. A zero value indicates the page is available for assignment. The array length is returned in the NLGPG= variable.						
'LTMASK=',msk	Large page truncation mask.						
'MACHID=',id	Machine identifier.						
'MACHINE=',flg	Hardware type flag. The possible integer values returned are: <table> <tr> <td>3</td><td>STAR 100</td></tr> <tr> <td>4</td><td>CYBER 200/Model 203</td></tr> <tr> <td>5</td><td>CYBER 200/Model 205</td></tr> </table>	3	STAR 100	4	CYBER 200/Model 203	5	CYBER 200/Model 205
3	STAR 100						
4	CYBER 200/Model 203						
5	CYBER 200/Model 205						
'MARKER=',flg	Indicator written in the pack label to indicate pack usage at a particular autoloader.						
'MAXPAGE=',n	Maximum available page number.						
'MCLOCK=',clck	ASCII value of the master clock (Wyman clock). The value is in the format yymmddsspppp where yy is the year, mm is the month, dd is the day, ss is the second, and pppp is the decimal fraction of a second.						
'MILLSEC=',sec	Elapsed milliseconds since an arbitrary time base.						
'MLPG=',n	Large page limit for the overall machine.						
'MXBYCNT=',mb	Maximum bypass count for a job in the input queue. Refer to Job Scheduling in section 3 for more information.						
'MXMO=',mx	Integer indicating the percentage of memory overcommitment allowed. This value is used when determining if a job can be scheduled.						
'MXRR=',mxr	Maximum rerun time in system seconds. The combined time limits of all executing jobs cannot exceed this value.						
'NLGPG=',n	Number of large pages in the machine. This parameter must be specified if LGPG= or LPAV= are specified.						
'NOPAGE=',n	Number of pages in Page Table.						
'NUREG=',bit	Pack usage bits. bit must be a two-word array.						
'OCCPA=',n	Number of occupied small pages minus one.						
'OPRID=',un	Operator user number; the value is hexadecimal, right-justified with null fill.						
'OPTB2=',op	Copy virtual system option.						
'OPTCKSM=',op	Checksumming option.						
'OPTHBIT=',op	Record history option.						
'OPTLKVS=',op	Virtual system lock option.						
'OPTMCU=',op	MCU option.						
'OPTNODM=',op	No drum option.						
'OPTSPRG=',op	Special dedicated memory region option.						
'OPTTSBI=',op	Time stamp option.						

Figure 8-4. Q5DCDMSC Call Format (Sheet 2 of 3)

'OPTTYOT=',n	Operator output TTY number.
'OPZIP=',zip	Zip code of the first-level station used to communicate with the AUTOCON module. The value is hexadecimal and right-justified.
'P=',db	Descriptor block number currently running the alternator.
'PP=',db	Descriptor block number for alternator AA.
'PAGRE=',n	Number of pages reserved by PAGER.
'PFADDR=',adr	Physical disk address of paging file.
'PFLN=',len	Length in small pages of paging file.
'PFUNIT=',unt	Physical unit of paging file.
'PFZIP=',zip	Zip code of paging file.
'PUREG=',unt	Physical unit numbers. unt must be a two-word array.
'RDB=',db	Descriptor block number for which PAGER is reserved.
'RECOVFG=',n	Number of recoveries since last deadstart.
'RVER=',ver	Version number of the resident system.
'SHPAR=',ent	Shift count for large pages.
'SMPLG=',n	Number of small pages per large page.
'STATUS=',stat	Status code. Possible values: 0 through 299.
'STMM=',adr	Starting address of minus page table.
'SYSID=',id	System ID; five alphanumeric characters, left-justified, blank fill.
'TIME=',clk	Microsecond clock.
'TLSBU=',tl	Units in which the job time limit is measured. SIL returns one of the following binary values.
	0 System seconds.
	1 SBUs.
'VPTI=',vpt	Virtual Process Table index.
'VSSADDR=',adr	Physical disk address of virtual system recovery file.
'VSSLEN=',len	Length of virtual system recovery file.
'VSSUNIT=',unt	Unit number for virtual system recovery file.
'VSSZIP=',zip	Zip code of station for virtual system recovery file.
'VVER=',ver	Version number of virtual system.
'ZIPREG=',zip	Zipcodes of stations. zip must be a four-word array.

Figure 8-4. Q5DCDMSC Call Format (Sheet 3 of 3)

Q5DCDPFI-DECODE PACK FILE INDEX

The Q5DCDPFI subroutine (refer to figure 8-5) retrieves information from a permanent file index entry. The user must call the Q5GETPFI, Q5LFIPRI, Q5LFIPUB, or Q5LFIPOL subroutine to get a copy of the file index entry before issuing the Q5DCDPFI call.

SIL returns PFI information according to the parameters specified on the Q5DCDPFI call. The user must specify at least one return parameter (other than ERRLEN=, ERRMSG=, and STATUS=). All information returned (except the file name and error message) is in binary, rather than character, format.

Call Format

CALL Q5DCDPFI(optional parameters)

NOTE

The user must specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

'ENTRY=',n	Relative entry number (beginning with 1) of the file index entry to be decoded. If ENTRY= is omitted, SIL uses the entry number specified on the previous Q5DCDPFI call plus one. If the task has not previously called the Q5DCDPFI routine, the default is 1. The ENTRY= parameter and the MYFILE= parameter are mutually exclusive.
'MYFILE=',ary	Sixteen-word array containing a user-supplied file index entry. The array must begin on a word boundary. If MYFILE= is omitted, SIL decodes the file index entry obtained by the last Q5GETPFI, Q5LFIPOL, Q5LFIPRI, or Q5LFIPUB call and specified by the ENTRY= parameter. The MYFILE= parameter and the ENTRY= parameter are mutually exclusive.
'MYLEN=',len	Length of the array specified by the MYFILE= parameter. MYLEN= is required if MYFILE= is specified. The length specified must be 16.
'STLEN=',len	Segment table length in words. (The current table length is 4.) STLEN= is required if the SEGADR= or SEGLE= return parameters are specified.

Return Parameters

'ACS=',acs	File access permission. SIL returns one of the following ASCII values. NO No access. R Read access. RW Read and write access. T Write temporary access. W Write access.
'ACT=',act	Number of active I/O connectors for the file.
'ATSUF=',suf	Suffix under which file is attached. SIL can return the following ASCII values. blank Not attached. A Attached to suffix A. B Attached to suffix B. C Attached to suffix C. D Attached to suffix D.
'AWW=',cnt	Count of the active users with write access to the file.
'BT=',bt	Blocking type. SIL returns one of the following ASCII values. blank Non-SIL file. C Fixed character count.

Figure 8-5. Q5DCDPFI Call Format (Sheet 1 of 5)

'CKSF'X=',suf	Checkpoint suffix (ASCII, left-justified, blank-filled, eight bytes).		
	blank File does not belong to a checkpointed job. A Suffix A. B Suffix B. C Suffix C. D Suffix D.		
'CM=',cc	Code conversion to be performed at the access station. SIL can return one of the following ASCII values.		
	BI Binary. DI Display code (64-character set). EC Extended display code (128-character set).		
'CONT=',con	File contiguity as set when the file was created. SIL returns one of the following ASCII values.		
	Y File is contiguous. N File is not contiguous.		
'DC=',de	Disposition code. If DC= is omitted, SIL assumes the file is a scratch file. SIL returns one of the following ASCII values.		
	IN Input for batch processing at the access station. LR Print on 580-12 line printer at link station. LS Print on 580-16 line printer at link station. LT Print on 580-20 line printer at link station. NONE No disposition code set. PF Store as a permanent file at the access station. PR Print on any available printer. PU Punch. P1 Print on 501 line printer at link station. P2 Print on 512 line printer at link station. SC Scratch file (destroyed at end of task).		
'DEN=',den	Tape density. SIL returns one of the following ASCII values.		
	200 200 bpi. 556 556 bpi. 800 800 bpi. 1600 1600 bpi.		
'DI=',di	Delivery information for the access station (one to eight ASCII characters.).		
'DOLA=',dat	Date of last access to the file returned in the following format.		
	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 5px;">yy7</td> <td style="padding: 5px;">dddg</td> </tr> </table>	yy7	dddg
yy7	dddg		
	yy Last two digits of the year. ddd Number of days since the beginning of the year, 1 through 366.		
'DOLM=',dat	Date of last open request to this file with write access. The date is returned in the same format as the date returned by the DOLA= parameter.		
'DORG=',dat	Date this file was originated. SIL returns the date in the same format as the date returned with the DOLA= parameter.		
'DT=',dt	File device. SIL returns one of the following ASCII values.		
	MS Mass storage. MT 7-track tape. NT 9-track tape.		
'DUP=',dup	Duplicate file name flag. It is set to #D if a duplicate file entry exists; otherwise, it is set to 1.		

Figure 8-5. Q5DCDPFI Call Format (Sheet 2 of 5)

'EC=',ec	Print or punch representation of the file. SIL returns one of the following ASCII values.
	26 O26 punch format. 29 O29 punch format. 80 80-column binary punch format. B4 48-character BCD print train. B6 64-character BCD print train. A4 48-character ASCII print train. A6 64-character ASCII print train. A9 95-character ASCII print train. NONE No external characteristics set.
'ERRLEN=',len	Length of the error message in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'EXT=',ext	Extension permission flag as set when the file was created. SIL can return one of the following ASCII values.
	Y File is extendible. N File is not extendible.
'FACT=',acct	ASCII account number (eight bytes, left-justified with blank fill).
'FC=',fc	File category. SIL returns one of the following ASCII values.
	B Batch input file. N Not defined. S System-generated drop file. U User file.
'FG=',fg	File acquisition method. SIL returns one of the following ASCII values.
	Y The user created the file. N The user was given the file.
'FI=',fi	Privileged task flag. SIL returns one of the following ASCII values.
	Y Task is privileged. N Task is not privileged.
'FISTID=',id	Terminal identifier for the coupling station (ASCII, left-justified, two bytes).
'FO=',fo	File organization. SIL returns one of the following ASCII values.
	B Unstructured binary. S Non-SIL structured. U Unstructured ASCII.
'GIVETU1=',un	Binary user number of the user who gave this file to the USER1 routine.
'HBA=',hba	Highest byte accessed plus one.
'IC=',ic	File format. SIL can return one of the following ASCII values.
	AS 8-bit ASCII; ANSI carriage control. BI Binary. PA or 0 8-bit ASCII; ASCII carriage control. None No internal characteristic set.
'LB=',lb	Tape label format. SIL returns one of the following ASCII values.
	L ANSI labeled tape. U Unlabeled tape.
'LFN=',lfn	File name (eight bytes, left-justified ASCII).

Figure 8-5. Q5DCDPFI Call Format (Sheet 3 of 5)

'LOCAL=',loc	Indicates that the file is local or permanent. SIL returns one of the following ASCII values. N Not local (permanent). Y Local.
'LODLEN=',len	Length, in 512-word blocks, of the program's drop file. If the file is not a virtual code file, this field is zero.
'MNR=',mnr	Minimum record length in bytes.
'MXR=',mxr	Maximum record length in bytes.
'NAC=',nac	Access station area code.
'ORGOWNR=',un	User number of the originator (binary).
'ORIGLEN=',len	Original file length, in 512-word blocks, requested when file created.
'OT=',ot	Origin type of a file destined for the access station. SIL returns one of the following ASCII values. B Batch origin. E Remote batch origin. I Interactive origin.
'OWNDIV=',div	Division code (ASCII, four bytes, left-justified).
'PC=',pc	Padding character (ASCII, one byte, left-justified).
'PNO=',pno	Tape file position number.
'PO=',cnt	Privileged open count.
'POOL=',pool	Pool name (ASCII, left-justified, eight bytes).
'PTRPFIL=',pfi	PFI entry.
'REF=',n	Number of times the file has been opened.
'RP=',rp	Retention period of the file in days (integer).
'RERUN=',flg	Rerun flag (applies to batch input files only). SIL returns one of the following ASCII values. Y Rerun batch job. N Do not rerun batch job.
'RMK=',rmk	Record mark character (ASCII, left-justified, one byte).
'RT=',rt	Record type. SIL returns one of the following ASCII values. F ANSI fixed length. R Record mark delimited. U Undefined structure. W Control word delimited.
'SADDR=',adr	Disk address of the first block of the file.
'SEGADR=',adr	Four-word array in which SIL returns the file segment addresses, one address per word. The array must begin on a word boundary. If SEGADR= is specified, the STLEN= parameter is required.
'SEGLN=',len	Array in which SIL returns the file segment lengths, one length per word. The array must begin on a word boundary. If SEGLN= is specified, the STLEN= parameter is required.
'SFO=',sfo	SIL file organization. SIL returns the following ASCII value. S Sequential

Figure 8-5. Q5DCDPFI Call Format (Sheet 4 of 5)

'SHACC=',acc	Shared access permission for pool files. SIL returns one of the following ASCII values.
	NO No access. R Shared read access. RW Shared read and write access. W Shared write access.
'SLEV=',sl	Security level. SIL returns a value between 0 and 255.
'ST=',st	Site identifier. (ASCII, three bytes, left-justified). Except for the following identifiers, the installation determines the site identifiers.
	AST Access station. URS Unit record station.
'STATUS=',stat	Status code. SIL returns one of the following values: 0 through 199, 261, 504, 505.
'TID=',tid	Terminal identifier for the access station (ASCII, left-justified, seven bytes).
'TLR=',tlr	Time of last open request. SIL returns an integer representing the system clock time, in seconds since midnight, when the file was last opened.
'TOLM=',time	Time of last write access. SIL returns an integer representing the system clock time in seconds since midnight, when the file was last opened for write access.
'TORG=',time	File creation time. SIL returns an integer representing the system clock time, in seconds since midnight, when the file was created.
'TPM=',tpm	Tape mode. SIL returns one of the following ASCII values.
	ASC Unformatted binary; 8-bit ASCII code (7- or 9-track tape). AS6 Six-bit ASCII code (7-track tape). BCD External BCD code (7-track tape). BIN Unformatted binary; 8-bit ASCII code (7- or 9-track tape).
'TYPE=',typ	File type. SIL returns on of the following ASCII values.
	PD Physical data file. VC Virtual code file.
'UNIT=',unt	Logical unit number of the disk on which the file resides.
'USER=',un	User number (binary).
'VRI=',vri	Index into Variable Rate Accounting table (applies to virtual code files only).
'VSN=',vsn	Volume serial number (ASCII, left-justified, six bytes).
'WLEN=',len	File length in 512-word blocks.
'ZIP=',zip	Zip for the site identifier specified by the ST= parameter.

Figure 8-5. Q5DCDPFI Call Format (Sheet 5 of 5)

Q5DCDPLB-DECODE PACK LABEL

The Q5DCDPLB subroutine (refer to figure 8-6) retrieves information from a copy of a pack label. The user must issue a Q5GETPFI call to get a copy of the pack label before issuing the Q5DCDPLB call.

SIL returns the information specified by the return parameters on the call. The user must specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=). All information is returned in binary, rather than character, format.

Call Format

CALL Q5DCDPLB(parameters)

NOTE

The user must specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

None.

Return Parameters

'COM1=',m1	Three words describing the machine, serial number, and other information set by the NAMEPACK system routine.
'COM2=',m2	
'COM3=',m3	
'CREATE=',dat	Creation date (ASCII characters in the format mm.dd.yy).
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array in which SIL returns an error message.
'EXPIRE=',dat	Expiration date (ASCII characters in the format mm.dd.yy).
'LABEL=',adr	Disk block address of this label.
'PFIE=',n	Entry number of this entry within the pack file index, counting from zero.
'PFIL=',len	Original length of the pack file index.
'PFILOC=',adr	Disk block address of the first block of the permanent file index.
'PN=',pn	Pack name (ASCII, left-justified, six bytes).
'SERIES=',s	The ASCII characters, "Δ1" (two bytes, binary value #2031).
'STATUS=',stat	Status code. Possible values: 0 through 202, 250, 505.
'TYPE=',typ	Type of disk pack. The possible values returned are #841, #844, and #819.
'UPDATE=',dat	Date of the last update of the disk (ASCII characters in the format mm.dd.yy).
'VOLN=',vol	Volume field (ASCII, left-justified, eight bytes).

Figure 8-6. Q5DCDPLB Call Format

Q5DESBIF-DESTROY BATCH INPUT FILE

The Q5DESBIF subroutine (refer to figure 8-7) requests the system to destroy the specified batch input file if the system fails.

Q5DESBIF uses the Miscellaneous system message.

Example:

The following FORTRAN source lines request the system to destroy the batch input file if the system fails. The name of the batch input file is obtained from a copy of its file index entry via calls to Q5LFIPRI and Q5DCDPFI.

```
CHARACTER*8 LFN
      .
      .
      .
CALL Q5LFIPRI('BATCH','ATTACHED')
CALL Q5DCDPFI(LFN=,LFN)
CALL Q5DESBIF(LFN=,LFN)
      .
      .
      .
```

Q5DISAMI-DISABLE MESSAGE INTERRUPTS

The Q5DISAMI subroutine (refer to figure 8-8) disables message interrupts to the task.

Q5DISAMI uses the Program Interrupt system message. Message interrupt processing is described in the Program Interrupt system message description in volume 2.

Q5DISATI-DISABLE ABNORMAL TERMINATION CONTROL

The Q5DISATI subroutine (refer to figure 8-9) disables the abnormal termination control feature described under Abnormal Termination Control in section 3.

Q5DISATI uses the Abnormal Termination Control system message.

Call Format

CALL Q5DESBIF('LFN=',lfn,optional parameters)

Calling Parameters

'LFN=',lfn Name of the batch input file to be destroyed. The name must be left-justified with blank fill in a full word on a word boundary. This is a required parameter. The user determines the name of the batch input file by specifying the LFN= parameter on a Q5DCDPFI call.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).
'ERRMSG=',msg 80-byte array to which SIL returns an error message.
'STATUS=',stat Status code. Possible values: 0 through 299, 400.

Figure 8-7. Q5DESBIF Call Format

Call Format

CALL Q5DISAMI(optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299.

Figure 8-8. Q5DISAMI Call Format

Call Format

CALL Q5DISATI(optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299.

Figure 8-9. Q5DISATI Call Format

Q5DMPACT-DUMP CUMULATIVE ACCOUNTING BUFFER

The Q5DMPACT subroutine (refer to figure 8-10) dumps the cumulative accounting file to permanent storage and terminates the temporary file. Only a privileged user can issue the Q5DMPACT call.

Q5DMPACT uses the Accounting Communication system message.

Q5ENAMI-ENABLE MESSAGE INTERRUPTS

The Q5ENAMI subroutine (refer to figure 8-11) enables messages to interrupt the task. When the task is interrupted, the specified interrupt subroutine is executed.

To return control to the interrupted task, the interrupt subroutine must issue a Q5RFI call.

Q5ENAMI uses the Program Interrupt system message. Message interrupt processing is described in the Program Interrupt system message description in volume 2.

Example:

The following FORTRAN source lines enable message interrupts and specify MISUB as the interrupt subroutine.

```
EXTERNAL MISUB
```

```
CALL Q5ENAMI('SUBNAME=',MISUB)
```

Call Format

CALL Q5DMPACT(optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).
'ERRMSG=',msg 80-byte array to which SIL returns an error message.
'STATUS=',stat Status code. Possible values: 0 through 299.

Figure 8-10. Q5DMPACT Call Format

Call Format

CALL Q5ENAMI('SUBNAME=',sub,optional parameters)

Calling Parameters

'SUBNAME=',sub Name of the user's interrupt subroutine. The subroutine (declared external in the calling program) gains control if a message interrupt occurs. This is a required parameter.
'TERMINAL' Indicates that the user interrupts the program with terminal messages preceded by the left-justified characters (sc)I, where (sc) is a special character defined by the installation (refer to Request Lines in section 3). If 'TERMINAL' is omitted, all messages from a terminal interrupt the program.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).
'ERRMSG=',msg 80-byte array to which SIL returns an error message.
'STATUS=',stat Status code. Possible values: 0 through 299, 380.

Figure 8-11. Q5ENAMI Call Format

Q5ENATI-ENABLE ABNORMAL TERMINATION CONTROL

The Q5ENATI subroutine (refer to figure 8-12) enables the abnormal termination control feature described under Abnormal Termination Control in section 3.

Q5ENATI uses the Abnormal Termination Control system message.

Example:

The following FORTRAN source lines enable abnormal termination control and specify ATSUB as the interrupt subroutine.

```
EXTERNAL ATSUB
      .
      .
      .
CALL Q5ENATI('SUBNAME=',ATSUB)
```

Q5GETACT-GET RESOURCE USAGE STATISTICS

The Q5GETACT subroutine (refer to figure 8-13) obtains user accounting information. The user must specify at least one return parameter (other than the ERRLEN=, ERRMSG=, and STATUS= parameters). The information returned (except the error message) is in binary, rather than character, format. Q5GETACT uses the User/Accounting Communication system message.

Q5GETCTS-GET CONTROLLEE TERMINATION STATUS

The Q5GETCTS subroutine (refer to figure 8-14) gets the termination status of the task's controllee. Q5GETCTS also returns the system return code for the controllee.

Q5GETCTS uses the Miscellaneous system message.

Call Format

CALL Q5ENATI('SUBNAME=',sub,optional parameters)

Calling Parameters

'ERRLIM=',lim The maximum number (1 to 256) of error recoveries (excluding time limit errors). When this limit is exceeded, abnormal termination control aborts the task. If lim exceeds 256, the value of its lowest 8 bits is used (no error is recorded). If ERRLIM=,lim is omitted, the limit is 25 recoveries.

'SUBNAME=',sub The name of the user's interrupt subroutine or an entry point within the user's subroutine. Control transfers to the entry point if a predefined system fatal error occurs. The user must declare the subroutine as external in the calling program. This is a required parameter.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299, 380, 381.

Figure 8-12. Q5ENATI Call Format

Call Format

CALL Q5GETACT(parameters)

NOTE

The user must specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

None.

Return Parameters

'CPU TIME=' ,tim	User execution CPU time in microseconds.
'CWSSIZ=' ,siz	Current working size.
'DPLPEXP=' ,n	Number of large page explicit reads and writes to or from mass storage.
'DPLPFLT=' ,n	Number of mass storage reads due to large page faults.
'DPLPIMP=' ,n	Number of large page implicit writes to mass storage.
'DPSPEXP=' ,n	Number of small page explicit reads and writes to and from mass storage.
'DPSPFLT=' ,n	Number of mass storage reads due to small page faults.
'DPSPIMP=' ,n	Number of small page implicit writes to mass storage.
'DPXFEXP=' ,n	Number of sectors transferred to mass storage for explicit reads and writes.
'DPXFIMP=' ,n	Number of sectors transferred to mass storage for implicit writes.
'ERRLEN=' ,len	Length in bytes of the error message.
'ERRMSG=' ,msg	80-byte array to which SIL returns an error message.
'MEMUSE=' ,usg	Memory usage. At the end of each account period, (current working set size) user CPU time for current accounting period is computed and added to a running total kept in this field.
'MTACCES=' ,n	Number of magnetic tape reads and writes.
'MTNONIO=' ,n	Number of non-I/O magnetic tape operations.
'MTXFER=' ,n	Number of 16-bit bytes transferred to and/or from magnetic tape.
'SYSTIME=' ,tim	System CPU execution time in microseconds.
'STATUS=' ,stat	Status code. SIL returns one of the following values: 0 through 202, 250, 261.
'SYSCHRG=' ,n	Number of 16-bit bytes transferred to or from tape files.
'USRCHRG=' ,sbu	System billing units (real value).
'VSCALL=' ,n	Number of virtual system user calls made.
'WS2SM=' ,n	Cumulative CPU time in microseconds that this task's working set limit appeared to be too small.

Figure 8-13. Q5GETACT Call Format

Call Format

CALL Q5GETCTS($\left\{ \begin{array}{l} \text{'CTS='}, \text{cts} \\ \text{'RETCODE='}, \text{code} \end{array} \right\}$, optional parameters)

Calling Parameters

None.

Return Parameters

'CTS=',cts Controllee termination status. A mnemonic is returned left-justified with blank fill. The variable must be a full word on a word boundary. SIL returns one of the following ASCII values.

- AB Controllee aborted.
- OB Operator transferred control to the end-of-job card.
- OD Operator dropped job.
- OE Operator transferred control to the EXIT card.
- SA Controllee still active.
- TN Controllee terminated; files were not saved.
- TS Controllee terminated; files were saved.
- UB User entered terminal message transferring control to the EXIT card.

CTS= must be specified if RETCODE= is omitted.

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'RETCODE=',ret System return code. A mnemonic is returned left-justified with blank fill. The variable must be a full word on a word boundary. SIL returns one of the following ASCII values.

- blank No error.
- ERROR Warning error (nonfatal).
- FATAL Fatal error.

RETCODE= must be specified if CTS= is omitted.

'STATUS=',stat Status code. Possible values: 0 through 299

Figure 8-14. Q5GETCTS Call Format

Q5GETIIP-GET INVISIBLE PACKAGE

The Q5GETIIP subroutine (refer to figure 8-15) gets a copy of the task's invisible package after the task has been interrupted. The invisible package contains the address and control information required to continue execution of the task. The format of the invisible package is described in appendix E of volume 2.

Q5GETIIP uses the Miscellaneous system message.

Q5GETIRF-GET REGISTER FILE

The Q5GETIRF subroutine (refer to figure 8-16) gets a copy of the task's register file after the task has been interrupted. The register file consists of the contents of the 256 CYBER 200 hardware registers when the task is interrupted. The register file is saved when the job is interrupted. The format of the register file is described in appendix D of volume 2.

Q5GETIRF uses the Miscellaneous system message.

Call Format

CALL Q5GETIIP('INVPACK=',inv,optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'INVPACK=',inv 40-word array in which SIL returns the invisible package. This is a required parameter.

'STATUS=',stat Status code. Possible values: 0 through 299, 383.

Figure 8-15. Q5GETIIP Call Format

Call Format

CALL Q5GETIRF('REGFILE=',rf,optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'REGFILE=',rf 256-word array in which SIL returns the register file. This is a required parameter.

'STATUS=',stat Status code. Possible values: 0 through 299, 383.

Figure 8-16. Q5GETIRF Call Format

Q5GETLP-GET LARGE PAGE LIMITS

The Q5GETLP subroutine (refer to figure 8-17) gets the large page limits for the task. The maximum large page limit is set by either the job RESOURCE statement, the task execute line, or an installation-defined default value. The current large page limit is either the maximum large page limit or the limit specified on a previous SET statement or Q5SETLP call.

Q5GETLP uses the Process System Parameter system message.

The following call requests that SIL return the maximum large page limit in variable LPAGES and the current large page limit in variable NPAGES.

```
CALL Q5GETLP('NLP=',LPAGES,'RLP=',NPAGES)
```

Q5GETMCE-GET MESSAGE FROM CONTROLLEE

The Q5GETMCE subroutine (refer to figure 8-18) obtains a message from the task's controllee.

Q5GETMCE uses the Get Message From Controllee system message.

Call Format

Q5GETLP ('NLP=',nlp,optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'NLP=',nlp	Integer variable in which SIL returns the maximum large page limit for the task.
'RLP=',rlp	Integer variable in which SIL returns the current large page limit for the task.
'STATUS=',stat	Status code. Refer to Error Processing in section 8 for more information.

Figure 8-17. Q5GETLP Call Format

Call Format

CALL Q5GETMCE('MSG=',msg,optional parameters)

Calling Parameters

'LEN=',len	Message buffer length in bytes. If LEN= is omitted, SIL assumes an 80-byte message buffer.
'NULLFILL'	Indicates the fill character used if STD or SYD is specified. If NULLFILL is specified, binary zero is used. Otherwise, blank fill is used. NULLFILL must be omitted if STD and SYD are omitted.
'RJUSTIFY'	Indicates justification of symbols if STD or SYD is specified. If RJUSTIFY is specified, symbols are right-justified. Otherwise, symbols are left-justified. RJUSTIFY must be omitted if STD and SYD are omitted.
'SAVE'	Indicates the system buffer space is to be saved. If SAVE is omitted, the buffer space is released.
'STD'	Indicates use of standard delimiters (period, blank, comma, slash, equal, plus, minus, left parenthesis, and right parenthesis). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and null-filled.
'SYD'	Indicates use of system delimiters (defined by an installation parameter). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and null-filled.

Return Parameters

'DB=',db	Descriptor block number of the controllee that sent the message.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'LEVEL=',lev	Level, within the controllee chain, of the controllee that sent the message.
'MSG=',msg	Array to receive (optionally edited) message. This is a required parameter.
'MSGLEN=',n	Number of bytes received.
'STATUS=',stat	Status code. Possible values: 0 through 202, 250, 261, 340, 341, 342, 344, 345.

Figure 8-18. Q5GETMCE Call Format

Q5GETMCR-GET MESSAGE FROM CONTROLLER

The Q5GETMCR subroutine (refer to figure 8-19) obtains a message from the task's controller.

If a controllee other than a batch processor or interactive processor controllee (level 3 or greater) issues a Q5GETMCR call when it has no message waiting for it, controllee execution is suspended and its controller executes until it sends a message to the controllee.

Q5GETMCR uses the Get Message From Controller system message.

Q5GETMOP-GET MESSAGE FROM OPERATOR

The Q5GETMOP subroutine (refer to figure 8-20) obtains a message from the operator.

Q5GETMOP uses the Get Message From Operator system message.

Call Format

CALL Q5GETMCR('MSG=',msg,optional parameters)

Calling Parameters

'LEN=',len	Message buffer length in bytes. If LEN= is omitted, SIL assumes an 80-byte message buffer.
'NULLFILL'	Indicates the fill character used if STD or SYD is specified. If NULLFILL is specified, binary zero is used. Otherwise, blank fill is used. NULLFILL must be omitted if STD and SYD are omitted.
'REJECT'	Indicates that SIL should return an error code if a message is not waiting. If REJECT is omitted, SIL suspends task execution until a message is available.
'RJUSTIFY'	Indicates justification of symbols if STD or SYD is specified. If RJUSTIFY is specified, symbols are right-justified. Otherwise, symbols are left-justified. RJUSTIFY must be omitted if STD and SYD are omitted.
'SAVE'	Indicates the system buffer space is to be saved. If SAVE is omitted, the buffer space is released.
'STD'	Indicates use of standard delimiters (period, blank, comma, slash, equal, plus, minus, left parenthesis, and right parenthesis). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and null-filled.
'SYD'	Indicates use of system delimiters (defined by an installation parameter). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and null-filled.

Return Parameters

'DB=',db	Descriptor block number of the controller that sent the message.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'LEVEL=',lev	Level within the controllee chain of the controller that sent the message.
'MSG=',msg	Array to receive (optionally edited) message. This is a required parameter.
'MSGLEN=',n	Number of bytes received.
'STATUS=',stat	Status code. Possible values: 0 through 202, 250, 261, 340 through 343.

Figure 8-19. Q5GETMCR Call Format

Call Format

CALL Q5GETMOP('MSG=',msg,optional parameters)

Calling Parameters

'LEN=',len	Message buffer length in bytes. If LEN= is omitted, SIL assumes an 80-byte message buffer.
'NULLFILL'	Indicates the fill character used if STD or SYD is specified. If NULLFILL is specified, binary zero is used. Otherwise, blank fill is used. NULLFILL must be omitted if STD and SYD are omitted.
'REJECT'	Indicates that SIL should return an error code is a message is not waiting. If REJECT is omitted, SIL suspends task execution until a message is available.
'RJUSTIFY'	Indicates justification of symbols if STD or SYD is specified. If RJUSTIFY is specified, symbols are right-justified. Otherwise, symbols are left-justified. RJUSTIFY must be omitted if STD and SYD are omitted.
'SAVE'	Indicates the system buffer space is to be saved. If SAVE is omitted, the buffer space is released.
'STD'	Indicates use of standard delimiters (period, blank, comma, slash, equal, plus, minus, left parenthesis, and right parenthesis). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and null-filled.
'SYD'	Indicates use of system delimiters (defined by an installation parameter). STD and SYD are mutually exclusive. If neither is specified, the message is unedited, left-justified, and null-filled.

Return Parameters

'DB=',db	Descriptor block number of the operator task.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'MSG=',msg	Array to receive (optionally edited) message. This is a required parameter.
'MSGLEN=',n	Number of bytes received.
'STATUS=',stat	Status code. Possible values: 0 through 202, 250, 261, 340 through 342.

Figure 8-20. Q5GETMOP Call Format

Q5GETMPG-GET MINUS PAGE

The Q5GETMPG subroutine (refer to figure 8-21) gets a copy of the task's minus page information. The minus page is used for program communication with the operating system as described in section 2 of volume 2.

Q5GETMPG uses the Miscellaneous system message.

SIL copies the pack label and file indices into a buffer it defines. The user retrieves information from the pack label and file indices with the Q5DCDPLB and Q5DCDPFI subroutines.

The buffer used by the Q5GETPFI routine is the same buffer used by the Q5LFIPRI, Q5LFIPOL, and Q5LFIPUB routines. A call to any of these routines overwrites the contents of the buffer.

Q5GETPFI-GET PACK LABEL AND FILE INDEX

The Q5GETPFI subroutine (refer to figure 8-22) gets a copy of the pack label and file indices from the specified disk pack. Only a privileged user can call Q5GETPFI.

Q5GETPFI uses the Get Pack Label and PFI system message for an unformatted PFI.

Call Format

CALL Q5GETMPG('MPAGE=',mpage, optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'MPAGE=',mp	1024-word array in which SIL returns information from the task's minus pages. The array must be on a word boundary. Word 513 contains the value #FFFF if a second minus page is not returned. This is a required parameter.
'STATUS=',stat	Status code. Possible values: 0 through 299.

Figure 8-21. Q5GETMPG Call Format

Call Format

CALL Q5GETPFI('PN=',pn,optional parameters)

Calling Parameters

'PN=',pn	Name of the pack from which SIL gets the pack label and file indices. This parameter is required.
----------	---

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'NFILES=',n	Number of file indices returned.
'STATUS=',stat	Status code. Possible values: 0 through 202, 250, 300, 310 through 312.

Figure 8-22. Q5GETPFI Call Format

Q5GETTL-GET TIME LIMIT

The Q5GETTL subroutine (refer to figure 8-23) gets the existing time limit of the task.

Q5GETTL uses the Miscellaneous system message.

- Source file name.
- Drop file name.
- Suffix.
- Level in the controllee chain.
- User number.
- Privileged status.

Q5GETTN - GET TASK ATTRIBUTES

The Q5GETTN subroutine (refer to figure 8-24) can get the following information about a task.

Q5GETTN uses the Miscellaneous system message.

Call Format

CALL Q5GETTL('OLDTIME=',time,optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).
'ERRMSG=',msg 80-byte array to which SIL returns an error message.
'OLDTIME=',tl Existing time limit. This parameter is required.
'STATUS=',st Status code. Possible values: 0 through 299.

Figure 8-23. Q5GETTL Call Format

Call Format

CALL Q5GETTN(parameters)

NOTE

The user must specify at least one return parameter (other than ERRLEN=, ERRMSG=, or STATUS=).

Calling Parameters

None.

Return Parameters

'BINARY=',lfn Source file name for the task, (ASCII, right-justified, with blank fill).
'DROPFIL=',lfn Drop file name for the task, (ASCII, right-justified, with blank fill).
'ERRLEN=',len Length of error message in bytes (integer).
'ERRMSG=',msg 80-byte array in which SIL returns an error message.
'LEVEL=',lev Level of this task in the controllee chain.
'PRIV=',prv Privileged user flag. SIL returns one of the following ASCII values.
 Y Privileged.
 N Nonprivileged.
'STATUS=',stat Status code. SIL returns one of the following values: 0 through 202, 250, 261.
'SUFFIX=',suf Suffix to which the task is attached. SIL returns the ASCII character A, B, C, or D.
'USER=',un User number (ASCII, right-justified, six bytes).

Figure 8-24. Q5GETTN Call Format

Q5GETUID-GET USER NUMBER

The Q5GETUID subroutine (refer to figure 8-25) gets the user number under which the job is executing and the amount of execution time available for completion of the job.

Q5GETUID uses the Miscellaneous system message.

Q5INIT-INITIALIZE CONTROLLEE

The Q5INIT subroutine (refer to figure 8-26) initializes a controllee.

Q5INIT uses the Initialize Controllee system message.

Call Format

CALL Q5GETUID({ 'ACCTIME=',time
'USER=',user } ,optional parameters)

Calling Parameters

None.

Return Parameters

'ACCTIME=',sec Number of microseconds in the user's bank account. ACCTIME= is required if USER= is omitted.

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299.

'USER=',un User number (integer). The variable must be a full word on a word boundary. USER= is required if ACCTIME= is omitted.

Figure 8-25. Q5GETUID Call Format

Call Format

CALL Q5INIT('LFN=',lfn,optional parameters)

Calling Parameters

'LFN=',lfn Name of the controllee file to be initialized. This is a required parameter.

'TLIMIT=',tl Time limit for the controllee program in microseconds. If TLIMIT is omitted, SIL uses the time limit of the calling program.

'WAIT' Indicates that after the controllee is initialized, the calling task is suspended and the controllee starts executing. If WAIT is omitted, the calling task continues execution after controllee initialization.

Return Parameters

'DB=',db Descriptor block number identifying the initialized controllee. Its descriptor block number could change if the controllee is disconnected.

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array in which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 202, 250, 261, 350 through 367.

Figure 8-26. Q5INIT Call Format

Q5INITCH-INITIALIZE CONTROLLEE CHAIN

The Q5INITCH subroutine (refer to figure 8-27) initializes a chain of controllees. The calling task is the controller of the chain.

The maximum number of tasks in a controllee chain is nine. The calling task is already within the controllee chain. A user program is a controllee of the batch processor

or interactive processor; therefore, it can initiate a chain of no more than seven controllees.

Any error in the call prevents further initialization of the controllee chain.

Q5INITCH uses the Initialize Controllee Chain system message.

Call Format

CALL INITCH(LFN=',lfnlist','NTASKS=',n,optional parameters)

NOTE

The NTASKS= parameter specifies the number of controllee tasks. All other parameters must specify arrays with one word for each controllee task.

Calling Parameters

- 'DESTDOWN=',lev Array of integers corresponding to the files in the LFN= array. Each integer indicates the relative level of the controllee that is to receive the messages sent to this task by a controller (0 indicates the messages go to this task, 1 indicates the next lower level controllee, and so forth). If DESTDOWN= is omitted, SIL assumes 0 for each task.
- 'DESTUP=',lev Array of integers corresponding to the files in the LFN= array. Each integer indicates the relative level of the controller that is to receive messages sent to this task by a controllee. (0 indicates the messages go to this task, 1 indicates the next higher level controller, and so forth.) If DESTUP= is omitted, SIL assumes 0 for each task.
- 'LFN=',lfn Array of 1 through 7 ASCII filenames. The names are those of the controllee or drop files that comprise the controllee chain to be initiated. The order of the names in the array is the order of the tasks in the chain. This is a required parameter.
- 'NTASKS=',n Number of controllee tasks (1 through 7) to be initiated (number of entries in the LFN= array). This is a required parameter.
- 'TMLIMIT=',tl Array of integers corresponding to the files in the LFN= array. Each integer indicates the time limit in microseconds for that task. If TMLIMIT= is omitted, each task is given the time limit of its controller.

Return Parameters

- 'CEDB=',db Array of integers corresponding to the files specified in the LFN= array. Each integer returned is the descriptor block number of the corresponding task's controllee.
- 'CRDB=',db Array of integers corresponding to the files specified in the LFN= array. Each integer returned is the descriptor block number of the corresponding task's controller.
- 'DB=',db Array of integers corresponding to the files specified in the LFN= array. Each integer returned is the descriptor block number of the corresponding task.
- 'ERRLEN=',len Array of integers corresponding to the files in the LFN= array. Each integer returned is the error message length for the corresponding message in the ERRMSG= array.
- 'ERRMSG=',msg Array of 80-byte arrays corresponding to the files specified in the LFN= array. The error message corresponding to the file is returned in the appropriate 80-byte array.
- 'LEVEL=',lev Array of integers corresponding to the files specified in the LFN= array. Each integer returned is the absolute level of the corresponding controllee task.
- 'STATUS=',stat Array of status codes. Possible values: 0 through 299, 350 through 359, 367.

Figure 8-27. Q5INITCH Call Format

Q5LFIPOL-LIST POOL FILE INDICES

The Q5LFIPOL subroutine (refer to figure 8-28) gets a copy of the file index entries of a set of attached pool files. The set of files is determined by the qualifiers specified on the call. SIL copies the entries into a buffer it defines. The user calls the Q5DCDPFI subroutine to retrieve information from a file index entry.

The buffer used by the Q5LFIPOL routine is the same buffer used by the Q5GETPFI, Q5LFIPRI, and Q5LFIPUB routines. A call to any of these routines overwrites the contents of the buffer.

Q5LFIPOL uses the List Unformatted File Index system message.

Call Format

CALL Q5LFIPOL('POOLNAM=',pool,optional parameters)

Calling Parameters

'DC=',dc	Disposition code. File index entries are returned only for files with the specified disposition code. If DC= is omitted, the file disposition codes is not used to determine the set of files. The disposition codes are as follows. <table><tr><td>'IN'</td><td>Batch input to access station.</td></tr><tr><td>'LR'</td><td>Print on 580-12 line printer.</td></tr><tr><td>'LS'</td><td>Print on 580-16 line printer.</td></tr><tr><td>'LT'</td><td>Print on 580-20 line printer.</td></tr><tr><td>'PF'</td><td>Store as permanent file at access station.</td></tr><tr><td>'PR'</td><td>Print on any available line printer.</td></tr><tr><td>'PU'</td><td>Punch.</td></tr><tr><td>'P1'</td><td>Print on 501 line printer.</td></tr><tr><td>'P2'</td><td>Print on 512 line printer.</td></tr><tr><td>'SC'</td><td>Scratch file.</td></tr><tr><td>'*'</td><td>Any disposition code.</td></tr></table>	'IN'	Batch input to access station.	'LR'	Print on 580-12 line printer.	'LS'	Print on 580-16 line printer.	'LT'	Print on 580-20 line printer.	'PF'	Store as permanent file at access station.	'PR'	Print on any available line printer.	'PU'	Punch.	'P1'	Print on 501 line printer.	'P2'	Print on 512 line printer.	'SC'	Scratch file.	'*'	Any disposition code.
'IN'	Batch input to access station.																						
'LR'	Print on 580-12 line printer.																						
'LS'	Print on 580-16 line printer.																						
'LT'	Print on 580-20 line printer.																						
'PF'	Store as permanent file at access station.																						
'PR'	Print on any available line printer.																						
'PU'	Punch.																						
'P1'	Print on 501 line printer.																						
'P2'	Print on 512 line printer.																						
'SC'	Scratch file.																						
'*'	Any disposition code.																						
'EC=',ec	External characteristic. File index entries are returned only for files with the specified EC. If EC= is omitted, the file external characteristic is not used to determine the set of files. <table><tr><td>'26'</td><td>O26 punch format.</td></tr><tr><td>'29'</td><td>O29 punch format.</td></tr><tr><td>'80'</td><td>80-column binary punch format.</td></tr><tr><td>'B4'</td><td>BCD 48-character print train.</td></tr><tr><td>'B6'</td><td>BCD 64-character print train.</td></tr><tr><td>'A4'</td><td>ASCII 48-character print train.</td></tr><tr><td>'A6'</td><td>ASCII 64-character print train.</td></tr><tr><td>'A9'</td><td>ASCII 95-character print train.</td></tr><tr><td>'*'</td><td>Any external characteristic.</td></tr></table>	'26'	O26 punch format.	'29'	O29 punch format.	'80'	80-column binary punch format.	'B4'	BCD 48-character print train.	'B6'	BCD 64-character print train.	'A4'	ASCII 48-character print train.	'A6'	ASCII 64-character print train.	'A9'	ASCII 95-character print train.	'*'	Any external characteristic.				
'26'	O26 punch format.																						
'29'	O29 punch format.																						
'80'	80-column binary punch format.																						
'B4'	BCD 48-character print train.																						
'B6'	BCD 64-character print train.																						
'A4'	ASCII 48-character print train.																						
'A6'	ASCII 64-character print train.																						
'A9'	ASCII 95-character print train.																						
'*'	Any external characteristic.																						
'FNCOUNT=',n	Number of file names in the LFN= array. If LFN= is specified but FNCOUNT= is not, SIL assumes one file name specified.																						
'IC=',ic	File format. File index entries are returned only for files with the specified internal characteristic. If IC= is omitted, the file format is not used to determine the set of files. <table><tr><td>'AS'</td><td>8-bit ASCII format; ANSI carriage control.</td></tr><tr><td>'BI'</td><td>Binary format.</td></tr><tr><td>'PA'</td><td>8-bit ASCII format; ASCII carriage control.</td></tr><tr><td>'*'</td><td>Any internal characteristic.</td></tr></table>	'AS'	8-bit ASCII format; ANSI carriage control.	'BI'	Binary format.	'PA'	8-bit ASCII format; ASCII carriage control.	'*'	Any internal characteristic.														
'AS'	8-bit ASCII format; ANSI carriage control.																						
'BI'	Binary format.																						
'PA'	8-bit ASCII format; ASCII carriage control.																						
'*'	Any internal characteristic.																						
'LFN=',lfn	Array containing names of files for which PFI entries are obtained (ASCII, left-justified, with blank fill). If LFN= is omitted, file names are not used to determine the set of files.																						

Figure 8-28. Q5LFIPOL Call Format (Sheet 1 of 2)

'POOL=',pool	Name of the attached pool containing the files for which file index entries are to be obtained. POOL= is a required parameter.
'ST=',st	Site identifier. File index entries are returned only for files with the specified identifier. If ST= is omitted, the site identifier is not used to determine the set of files. Except for the following identifiers the installation determines the site identifiers.
'AST'	Access station.
'URS'	Unit record station.
'STRING'	Indicates that the entries in the LFN= arrays are strings. SIL returns file index entries for all files in the specified pool whose names begin with one of the strings. If STRING is omitted, SIL does not perform string matching.
'UNIT=',unt	Logical unit number containing files for which file index entries are to be obtained. If UNIT= is omitted, the unit number is not used to determine the set of files.
'ZIP=',zip	Zip code for the site identifier.

Return Parameters

'ERRLEN=',len	Length of the error message in bytes (integer).
'ERRMSG=',msg	80-byte array to which SIL returns an error message.
'NFILES=',n	Number of file index entries returned in the SIL-defined buffer.
'STATUS=',stat	Status code. SIL returns one of the following values: 0 through 202, 250, 261.

Figure 8-28. Q5LFIPOL Call Format (Sheet 2 of 2)

Q5LFIPRI-LIST PRIVATE FILE INDICES

The Q5LFIPRI subroutine (refer to figure 8-29) gets a copy of the file index entries of a set of private files. SIL copies the entries into a buffer it defines. The user calls the Q5DCDPFI subroutine to retrieve information from a file index entry.

The buffer used by the Q5LFIPRI routine is the same buffer used by the Q5GETPFI, Q5LFIPOL, and Q5LFIPUB routines. A call to any of these routines overwrites the contents of the buffer.

Q5LFIPRI uses the List Unformatted File Index system message.

Call Format

CALL Q5LFIPRI(optional parameters)

Calling Parameters

'ATTACHED'	Q5LFIPRI lists only those files attached to the suffix under which the calling task executes. If ATTACHED is omitted, Q5LFIPRI lists all files.																						
'BATCH'	Q5LFIPRI lists only batch input files. If BATCH is omitted, Q5LFIPRI lists all files.																						
'DC=',dc	Disposition code. File index entries are returned only for files with the specified disposition code. If DC= is omitted, the file disposition code is not used to determine the set of files. The disposition codes are as follows. <table><tr><td>'IN'</td><td>Batch input to access station.</td></tr><tr><td>'LR'</td><td>Print on 580-12 line printer.</td></tr><tr><td>'LS'</td><td>Print on 580-16 line printer.</td></tr><tr><td>'LT'</td><td>Print on 580-20 line printer.</td></tr><tr><td>'PF'</td><td>Store as permanent file at access station.</td></tr><tr><td>'PR'</td><td>Print on any available line printer.</td></tr><tr><td>'PU'</td><td>Punch.</td></tr><tr><td>'P1'</td><td>Print on 501 line printer.</td></tr><tr><td>'P2'</td><td>Print on 512 line printer.</td></tr><tr><td>'SC'</td><td>Scratch file.</td></tr><tr><td>'*'</td><td>Any disposition code.</td></tr></table>	'IN'	Batch input to access station.	'LR'	Print on 580-12 line printer.	'LS'	Print on 580-16 line printer.	'LT'	Print on 580-20 line printer.	'PF'	Store as permanent file at access station.	'PR'	Print on any available line printer.	'PU'	Punch.	'P1'	Print on 501 line printer.	'P2'	Print on 512 line printer.	'SC'	Scratch file.	'*'	Any disposition code.
'IN'	Batch input to access station.																						
'LR'	Print on 580-12 line printer.																						
'LS'	Print on 580-16 line printer.																						
'LT'	Print on 580-20 line printer.																						
'PF'	Store as permanent file at access station.																						
'PR'	Print on any available line printer.																						
'PU'	Punch.																						
'P1'	Print on 501 line printer.																						
'P2'	Print on 512 line printer.																						
'SC'	Scratch file.																						
'*'	Any disposition code.																						
'EC=',ec	External characteristic. File index entries are returned only for files with the specified EC. If EC= is omitted, the file external characteristic is not used to determine the set of files. <table><tr><td>'26'</td><td>O26 punch format.</td></tr><tr><td>'29'</td><td>O29 punch format.</td></tr><tr><td>'80'</td><td>80-column binary punch format.</td></tr><tr><td>'B4'</td><td>BCD 48-character print train.</td></tr><tr><td>'B6'</td><td>BCD 64-character print train.</td></tr><tr><td>'A4'</td><td>ASCII 48-character print train.</td></tr><tr><td>'A6'</td><td>ASCII 64-character print train.</td></tr><tr><td>'A9'</td><td>ASCII 95-character print train.</td></tr><tr><td>'*'</td><td>Any external characteristic.</td></tr></table>	'26'	O26 punch format.	'29'	O29 punch format.	'80'	80-column binary punch format.	'B4'	BCD 48-character print train.	'B6'	BCD 64-character print train.	'A4'	ASCII 48-character print train.	'A6'	ASCII 64-character print train.	'A9'	ASCII 95-character print train.	'*'	Any external characteristic.				
'26'	O26 punch format.																						
'29'	O29 punch format.																						
'80'	80-column binary punch format.																						
'B4'	BCD 48-character print train.																						
'B6'	BCD 64-character print train.																						
'A4'	ASCII 48-character print train.																						
'A6'	ASCII 64-character print train.																						
'A9'	ASCII 95-character print train.																						
'*'	Any external characteristic.																						
'FNCOUNT=',n	Number of file names in the LFN= array. If LFN= is specified but FNCOUNT= is not, SIL assumes one file name specified.																						
'IC=',ic	File format. File index entries are returned only for files with the specified internal characteristic. If IC= is omitted, the file format is not used to determine the set of files. <table><tr><td>'AS'</td><td>8-bit ASCII format; ANSI carriage control.</td></tr><tr><td>'BI'</td><td>Binary format.</td></tr><tr><td>'PA'</td><td>8-bit ASCII format; ASCII carriage control.</td></tr><tr><td>'*'</td><td>Any internal characteristic.</td></tr></table>	'AS'	8-bit ASCII format; ANSI carriage control.	'BI'	Binary format.	'PA'	8-bit ASCII format; ASCII carriage control.	'*'	Any internal characteristic.														
'AS'	8-bit ASCII format; ANSI carriage control.																						
'BI'	Binary format.																						
'PA'	8-bit ASCII format; ASCII carriage control.																						
'*'	Any internal characteristic.																						
'LFN=',lfn	Array containing names of files for which PFI entries are obtained (ASCII, left-justified, with blank fill). If LFN= is omitted, filenames are not used to determine the set of files.																						
'QF'	Q5LFIPRI lists only those batch input files that have not yet entered the input queue. If QF is omitted, the queue flag is not used to determine the set of files.																						

Figure 8-29. Q5LFIPRI Call Format (Sheet 1 of 2)

<p>'ST=',st</p> <p>'AST' Access station.</p> <p>'URS' Unit record station.</p> <p>'STRING'</p> <p>'UNIT=',unt</p> <p>'ZIP=',zip</p>	<p>Site identifier. File index entries are returned only for files with the specified identifier. If ST= is omitted, the site identifier is not used to determine the set of files. Except for the following identifiers, the installation determines the site identifiers.</p> <p>Indicates that the entries in the LFN= arrays are strings. SIL returns file index entries for all files in the specified pool whose names begin with one of the strings. If STRING is omitted, SIL does not perform string matching.</p> <p>Logical unit number containing files for which file index entries are to be obtained. If UNIT= is omitted, the unit number is not used to determine the set of files.</p> <p>Zip code for the site identifier.</p>
---	---

Return Parameters

<p>'ERRLEN=',len</p> <p>'ERRMSG=',msg</p> <p>'NFILES=',n</p> <p>'STATUS=',stat</p>	<p>Length of the error message in bytes (integer).</p> <p>80-byte array to which SIL returns an error message.</p> <p>Number of file index entries returned in the SIL-defined buffer.</p> <p>Status code. SIL returns one of the following values: 0 through 202, 261, 262, 300, 301.</p>
--	--

Figure 8-29. Q5LFIPRI Call Format (Sheet 2 of 2)

Q5LFIPUB-LIST PUBLIC FILE INDICES

The Q5LFIPUB subroutine (refer to figure 8-30) gets a copy of the file index entries of a set of public files. SIL copies the entries into a buffer it defines. The user calls the Q5DCDPFI subroutine to retrieve information from a file index entry.

The buffer used by the Q5LFIPUB routine is the same buffer used by the Q5GETPFI, Q5LFIPRI, and Q5LFIPOL routines. A call to any of these routines overwrites the contents of the buffer.

Q5LFIPUB uses the List Unformatted File Index system message.

Q5LSTBUT-LIST BANK UPDATE TABLE

The Q5LSTBUT subroutine (refer to figure 8-31) gets a copy of the Bank Update Table.

Q5LSTBUT issues the List System Table system message.

Q5LSTCH-LIST CONTROLLEE CHAIN

The Q5LSTCH subroutine (refer to figure 8-32) gets information about one or all tasks in the controllee chain. The subroutine can return the levels, descriptor block numbers, file names, drop file names, and time limits of the tasks in the chain. The descriptor block numbers are useful to identify a task directly in other SIL calls.

The number of words in the arrays specified to receive information must match the number of tasks in the chain (1 through 9).

Q5LSTCH uses the List Controllee Chain system message.

Call Format

CALL Q5LFIPUB(optional parameters)

Calling Parameters

'DC=',dc	Disposition code. File index entries are returned only for files with the specified disposition code. If DC= is omitted, the file disposition code is not used to determine the set of files. The disposition codes are as follows. <table><tr><td>'IN'</td><td>Batch input to access station.</td></tr><tr><td>'LR'</td><td>Print on 580-12 line printer.</td></tr><tr><td>'LS'</td><td>Print on 580-16 line printer.</td></tr><tr><td>'LT'</td><td>Print on 580-20 line printer.</td></tr><tr><td>'PF'</td><td>Store as permanent file at access station.</td></tr><tr><td>'PR'</td><td>Print on any available line printer.</td></tr><tr><td>'PU'</td><td>Punch.</td></tr><tr><td>'P1'</td><td>Print on 501 line printer.</td></tr><tr><td>'P2'</td><td>Print on 512 line printer.</td></tr><tr><td>'SC'</td><td>Scratch file.</td></tr><tr><td>'*'</td><td>Any disposition code.</td></tr></table>	'IN'	Batch input to access station.	'LR'	Print on 580-12 line printer.	'LS'	Print on 580-16 line printer.	'LT'	Print on 580-20 line printer.	'PF'	Store as permanent file at access station.	'PR'	Print on any available line printer.	'PU'	Punch.	'P1'	Print on 501 line printer.	'P2'	Print on 512 line printer.	'SC'	Scratch file.	'*'	Any disposition code.
'IN'	Batch input to access station.																						
'LR'	Print on 580-12 line printer.																						
'LS'	Print on 580-16 line printer.																						
'LT'	Print on 580-20 line printer.																						
'PF'	Store as permanent file at access station.																						
'PR'	Print on any available line printer.																						
'PU'	Punch.																						
'P1'	Print on 501 line printer.																						
'P2'	Print on 512 line printer.																						
'SC'	Scratch file.																						
'*'	Any disposition code.																						
'EC=',ec	External characteristic. File index entries are returned only for files with the specified EC. If EC= is omitted, the file external characteristic is not used to determine the set of files. <table><tr><td>'26'</td><td>O26 punch format.</td></tr><tr><td>'29'</td><td>O29 punch format.</td></tr><tr><td>'80'</td><td>80-column binary punch format.</td></tr><tr><td>'B4'</td><td>BCD 48-character print train.</td></tr><tr><td>'B6'</td><td>BCD 64-character print train.</td></tr><tr><td>'A4'</td><td>ASCII 48-character print train.</td></tr><tr><td>'A6'</td><td>ASCII 64-character print train.</td></tr><tr><td>'A9'</td><td>ASCII 95-character print train.</td></tr><tr><td>'*'</td><td>Any external characteristic.</td></tr></table>	'26'	O26 punch format.	'29'	O29 punch format.	'80'	80-column binary punch format.	'B4'	BCD 48-character print train.	'B6'	BCD 64-character print train.	'A4'	ASCII 48-character print train.	'A6'	ASCII 64-character print train.	'A9'	ASCII 95-character print train.	'*'	Any external characteristic.				
'26'	O26 punch format.																						
'29'	O29 punch format.																						
'80'	80-column binary punch format.																						
'B4'	BCD 48-character print train.																						
'B6'	BCD 64-character print train.																						
'A4'	ASCII 48-character print train.																						
'A6'	ASCII 64-character print train.																						
'A9'	ASCII 95-character print train.																						
'*'	Any external characteristic.																						
'FNCOUNT=',n	Number of file names in the LFN= array. If LFN= is specified but FNCOUNT= is not, SIL assumes one file name specified.																						
'IC=',ic	File format. File index entries are returned only for files with the specified internal characteristic. If IC= is omitted, the file format is not used to determine the set of files. <table><tr><td>'AS'</td><td>8-bit ASCII format; ANSI carriage control.</td></tr><tr><td>'BI'</td><td>Binary format.</td></tr><tr><td>'PA'</td><td>8-bit ASCII format; ASCII carriage control.</td></tr><tr><td>'*'</td><td>Any internal characteristic.</td></tr></table>	'AS'	8-bit ASCII format; ANSI carriage control.	'BI'	Binary format.	'PA'	8-bit ASCII format; ASCII carriage control.	'*'	Any internal characteristic.														
'AS'	8-bit ASCII format; ANSI carriage control.																						
'BI'	Binary format.																						
'PA'	8-bit ASCII format; ASCII carriage control.																						
'*'	Any internal characteristic.																						
'LFN=',lfn	Array containing names of files for which PFI entries are to be obtained (ASCII, left-justified with blank fill). If LFN= is omitted, file names are not used to determine the set of files.																						
'ST=',st	Site identifier. File index entries are returned only for files with the specified identifier. If ST= is omitted, the site identifier is not used to determine the set of files. Except for the following identifiers, the installation determines the site identifiers. <table><tr><td>'AST'</td><td>Access station.</td></tr><tr><td>'URS'</td><td>Unit record station.</td></tr></table>	'AST'	Access station.	'URS'	Unit record station.																		
'AST'	Access station.																						
'URS'	Unit record station.																						
'STRING'	Indicates that the entries in the LFN= arrays are strings. SIL returns file index entries for all files in the specified pool whose names begin with one of the strings. If STRING is omitted, SIL does not perform string matching.																						
'UNIT=',unt	Logical unit number containing files for which file index entries are to be obtained. If UNIT= is omitted, the unit number is not used to determine the set of files.																						
'ZIP=',zip	Zip code for the site identifier.																						

Figure 8-30. Q5LFIPUB Call Format (Sheet 1 of 2)

Return Parameters

'ERRLEN=',len Length of the error message in bytes (integer).
'ERRMSG=',msg 80-byte array to which SIL returns an error message.
'NFILES=',n Number of file index entries returned in the SIL-defined buffer.
'STATUS=',stat Status code. SIL returns one of the following values: 0 through 202, 261, 262, 300, 301.

Figure 8-30. Q5LFIPUB Call Format (Sheet 2 of 2)

Call Format

CALL Q5LSTBUT('BUT=',but,optional parameters)

Calling Parameters

None.

Return Parameters

'BUT=',but 32-word array in which SIL returns the Bank Update Table. The array must be a word boundary. This parameter is required.
'ERRLEN=',len Error message length in bytes (integer format).
'ERRMSG=',msg 80-byte array to which SIL returns an error message.
'STATUS=',stat Status code. Possible values: 0 through 299.

Figure 8-31. Q5LSTBUT Call Format

Call Format

CALL Q5LSTCH(optional parameters)

Calling Parameters

- 'CONTROLE' Indicates that SIL obtains information only on the task's controllee. The CONTROLE, CONTROLR, and PROGRAM parameters are mutually exclusive. If all three are omitted, SIL obtains information for all tasks in the controllee chain.
- 'CONTROLR' Indicates that SIL obtains information only on the task's controller. The CONTROLE, CONTROLR, and PROGRAM parameters are mutually exclusive. If all three are omitted, SIL obtains information for all tasks in the controllee chain.
- 'PROGRAM' Indicates that SIL obtains information only on the calling task. The CONTROLE, CONTROLR, and PROGRAM parameters are mutually exclusive. If all three are omitted, SIL obtains information for all tasks in the controllee chain.

Return Parameters

- 'BINARY=',lfn One- to nine-word array in which SIL returns the file names (in ASCII) of the controllees in the chain.
- 'CEDB=',db One- to nine-word array of integers corresponding to the file names in the BINARY= array. Each integer is the descriptor block number of the corresponding task's controllee.
- 'CRDB=',db One- to nine-word array of integers corresponding to the file names in the BINARY= array. Each integer is the descriptor block number of the corresponding task's controller.
- 'DB=',db One- to nine-word array of integers corresponding to the file names in the BINARY= array. Each integer is the descriptor block number of the corresponding task. If the task is interactive, #FF is returned. If the task is the batch processor, 1 is returned.
- 'DROPFIL=',lfn One- to nine-word array of file names (in ASCII) corresponding to the file names returned in the BINARY= array. Each file name is the name of the drop file for the corresponding task.
- 'ERRLEN=',len Error message length in bytes (integer format).
- 'ERRMSG=',msg 80-byte array to which SIL returns an error message.
- 'SUDB=',db Descriptor block number of the calling program.
- 'SULVL=',lev Level of the calling program.
- 'LEVEL=',lev One- to nine-word array of integers corresponding to the file names returned in the BINARY= array. Each integer is the level of the corresponding task.
- 'RNLVL=',n Number of levels the call returned in the LEVEL= array.
- 'STATUS=',stat Status code. Possible values:
0 through 299.
- 'TMLIMIT=',tl One- to nine-word array of integers corresponding to the file names returned in the BINARY= array. Each integer is the time limit in microseconds for the corresponding task.

Figure 8-32. Q5LSTCH Call Format

Q5LSTSTB-LIST STATISTICS BUFFER

The Q5LSTSTB subroutine (refer to figure 8-33) gets a copy of the Statistics Buffer. The Statistics Buffer is a system table that could contain system performance data.

Q5LSTSTB uses the List System Table system message.

Q5LSTTCB-LIST TIMECARD BUFFER

The Q5LSTTCB subroutine (refer to figure 8-34) gets a copy of the Timecard Buffer. The Timecard Buffer is a buffer that could contain accounting information.

Q5LSTTCB uses the List System Table system message.

Call Format

CALL Q5LSTSTB('STB=',stb,optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299.

'STB=',stb 100-word array in which SIL returns the Statistics Buffer. The array must be on a word boundary. This is a required parameter.

Figure 8-33. Q5LSTSTB Call Format

Call Format

CALL Q5LSTTCB('TCB=',tcb,optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer format).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299.

'TCB=',tcb 512-word array in which SIL returns the Timecard Buffer. The array must be on a word boundary. This is a required parameter.

Figure 8-34. Q5LSTTCB Call Format

Q5MSGCTR-MESSAGE CONTROL

The Q5MSGCTR subroutine (refer to figure 8-35) redirects messages sent to the task. Q5MSGCTR can direct messages from the task's controllee to one of the task's controllers and direct messages from the task's controller to one of the task's controllees (refer to table 8-1).

Q5MSGCTR uses the Message Control system message.

Q5RECALL-SUSPEND TASK EXECUTION

The Q5RECALL subroutine (refer to figure 8-36) suspends task execution for a specified length of time.

Q5RECALL uses the Recall system message.

Call Format

CALL Q5MSGCTR($\left\{ \begin{array}{l} \text{'CEDB='}, \text{db} \\ \text{'CRDB='}, \text{db} \end{array} \right\}$, optional parameters)

Calling Parameters

'CEDB=',db Descriptor block number identifying the controllee to receive messages sent to this task by its controller. Zero indicates the next level controllee. If CEDB= is omitted, CRDB= is required. SIL uses the rightmost eight bits of the value.

'CRDB=',db Descriptor block number identifying the controller to receive messages sent to this task by its controllee. 0 identifies the next level controller; #FF identifies the job control processor. If CRDB= is omitted, CEDB= is required. SIL uses the rightmost eight bits of the value.

'OFF' Ends message redirection set by previous Q5MSGCTR calls. The effect of the OFF parameter is summarized in table 8-1.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299, 321.

Figure 8-35. Q5MSGCTR Call Format

Call Format

CALL Q5RECALL(optional parameters)

Calling Parameters

'TIME=',sec Number of seconds (30 through 1800) that SIL suspends task execution. If TIME= is omitted, SIL suspends the task for 30 seconds.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299, 410.

Figure 8-36. Q5RECALL Call Format

TABLE 8-1. MESSAGE REDIRECTION

CEDB=db	CRDB=db	OFF	Effect
Specified	Omitted	Omitted	Turns on controller message redirection.
Omitted	Specified	Omitted	Turns on controllee message redirection.
Specified	Specified	Omitted	Turns on controllee and controller message redirection.
Specified	Omitted	Specified	Turns off controller message redirection.
Omitted	Specified	Specified	Turns off controllee message redirection.
Specified	Specified	Specified	Turns off controllee and controller message redirection.

Q5RFI-RETURN FROM INTERRUPT SUBROUTINE

The Q5RFI subroutine (refer to figure 8-37) returns control from an interrupt subroutine to the interrupted task. The user can choose one of the following processing options for the interrupted task.

- Abort at the point of the original interrupt.
- Continue processing at the point of the original interrupt.
- Continue processing at a specified entry point.

If the user chooses to abort processing or if the task aborts after control is returned from the interrupt subroutine, the user receives the normal dump and traceback information; however, the interrupt subroutine is not shown in the traceback information. If the user specifies the RFISUB= parameter on the Q5RFI call and the task subsequently aborts, the traceback information shows the RFISUB= entry point as being called from the original point of interrupt.

Q5RFI issues the Return From Interrupt system message.

Call Format

CALL Q5RFI(optional parameters)

Calling Parameters

'ABORT' Indicates the program should abort at the point of the original interrupt. The user must not specify both the ABORT and the RFISUB= parameters. If neither is specified, the program continues processing at the point of interruption.

'RFISUB=',sub Entry point name (in ASCII) where processing continues. The entry point must be declared external in the interrupted program. When a fatal error occurs, the point of interrupt appears to call the entry point. The user must not specify both the ABORT and RFISUB= parameters. If neither is specified, the program continues processing at the point of interruption.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0, 381, 420.

Figure 8-37. Q5RFI Call Format

Q5RUNBIF-RERUN BATCH INPUT FILE

The Q5RUNBIF subroutine (refer to figure 8-38) informs the system that the specified batch input file is to be rerun if the system fails.

Q5RUNBIF uses the Miscellaneous system message.

Example:

The following FORTRAN source line requests the system to rerun the batch input file if the system fails. The name of the batch input file is obtained from a copy of its file index entry via calls to Q5LFIPRI and Q5DCDPFI.

```
CHARACTER*8 LFN
CALL Q5LFIPRI('BATCH','ATTACHED')
CALL Q5DCDPFI('LFN=',LFN)
CALL Q5RUNBIF('LFN=',LFN)
```

Q5SETLP-CHANGE CURRENT LARGE PAGE LIMIT

The Q5SETLP subroutine (refer to figure 8-39) can change the current large page limit for the task. The specified current large page limit must not exceed the maximum large page limit for the job or task. The user can determine the maximum large page limit and the current large page limit with a Q5GETLP call.

If the task has more large pages allocated than the specified current large page limit, the contents of the excess large pages are immediately paged out of memory.

Q5SETLP uses the Process System Parameter system message.

Example:

The following call sets the current large page limit at six pages:

```
CALL Q5SETLP ('NLP=',6)
```

Call Format

CALL Q5RUNBIF('LFN=',lfn,optional parameters)

Calling Parameters

'LFN=',lfn Name of the batch input file to be rerun. The name must be left-justified with blank fill in a full word on a word boundary. This is a required parameter. The user can determine the name of the batch input file by specifying the LFN= parameter on a Q5DCDPFI call.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299, 400.

Figure 8-38. Q5RUNBIF Call Format

Call Format

Q5SETLP('NLP=',nlp, optional parameters)

Calling Parameter

'NLP=',nlp Current large page limit for task (decimal integer).

Return Parameters

'ERRLEN=',len Error message length in bytes (integer)

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Refer to Error Processing in section 8 for more information.

Figure 8-39. Q5SETLP Call Format

Q5SNDMCE-SEND MESSAGE TO CONTROLLEE

The Q5SNDMCE subroutine (refer to figure 8-40) sends a message to the calling task's controllee.

If the controllee is a compiled FORTRAN program that has been initialized but whose execution has not begun, the first message sent to the controllee must be for reassignment of the files named in the PROGRAM statement (refer to Execution-Time File Reassignment in the CYBER 200 FORTRAN Reference Manual).

Q5SNDMCE uses the Send Message to Controllee system message.

Q5SNDMCR-SEND MESSAGE TO CONTROLLER

The Q5SNDMCR subroutine (refer to figure 8-41) sends a message to the calling task's controller. If the controller is the batch processor, the message is written in the task's job dayfile.

Q5SNDMCR uses the Send Message to System Controller system message.

Call Format

CALL Q5SNDMCE('MSG=',msg,optional parameters)

Calling Parameters

'DB=',db	Descriptor block number identifying the controllee to receive the message. If DB= is omitted, the message goes to the next lower controllee in the chain.
'LEN=',len	Length in bytes of the message to be sent. If LEN= is omitted, SIL assumes the first character of the message is a delimiter and the message consists of the second character through the character preceding the next occurrence of the delimiter. SIL sends a maximum of 2000 bytes; if the message exceeds that length, SIL truncates it, but does not return an error.
'MSG=',msg	Message to be sent. This parameter is required.
'REJECT'	Indicates that SIL should return an error code if the message cannot be sent immediately. If REJECT is omitted, this message replaces any existing message.

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array in which SIL returns an error message.
'STATUS=',stat	Status code. Possible values: 0 through 202, 250, 261, 320, 321, 325.

Figure 8-40. Q5SNDMCE Call Format

Call Format

CALL Q5SNDMCR('MSG=',msg,optional parameters)

Calling Parameters

'DB=',db	Descriptor block number identifying the controller to receive the message. If DB= is omitted, the message goes to the next higher controller in the chain.
'LEN=',len	Length in bytes of the message to be sent. If LEN= is omitted, SIL assumes the first character of the message is a delimiter and the message consists of the second character through the character preceding the next occurrence of the delimiter. SIL sends a maximum of 2000 bytes after removing the delimiters. If the message exceeds that length, SIL truncates it, but does not return an error.
'MSG=',msg	Message (1 through 2000 bytes). This is a required parameter.
'REJECT'	Indicates that if the message would replace an existing message, SIL suspends task execution until the message can be sent. If REJECT is omitted, this message replaces any existing message. If REJECT is specified, RETURN must be omitted.
'RETURN'	Indicates that SIL should return an error code if the message cannot be sent immediately. If RETURN is omitted, this message replaces any existing message. If RETURN is specified, REJECT must be omitted.

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	80-byte array in which SIL returns an error message.
'STATUS=',stat	Status code. Possible values: 0 through 202, 250, 261, 320 through 324.

Figure 8-41. Q5SNDMCR Call Format

Q5SNDMDF-SEND MESSAGE TO DAYFILE

The Q5SNDMDF subroutine (refer to figure 8-42) sends a message to the job's dayfile.

If the specified message is longer than 2000 bytes, SIL truncates the message without returning an error code.

If this message fills the dayfile, the system replaces it with the message DAYFILE FULL. After that message is written, no more messages can be written on the job dayfile.

Illegal characters (#00 through #1E, #7F through #FF) are changed to blanks, except for #000A, which is changed to #201F. The system adds an end-of-line character (#1F) if none is specified.

Q5SNDMDF uses the Send Message to Dayfile system message.

Example:

The following Q5SNDMDF call sends a message to the dayfile.

```
CHARACTER * 80 ERRARY(80)
CALL Q5SNDMDF('MSG=', 'THIS IS A MESSAGE',
+'LEN=',17)
```

Q5SNDMJC-SEND MESSAGE TO JOB CONTROLLER

The Q5SNDMJC subroutine (refer to figure 8-43) sends a message to the calling task's job controller (batch processor or virtual system interactive processor). If the job controller is the batch processor, the message is written in the job dayfile.

Q5SNDMJC uses the Send Message to Controller system message.

Q5SNDMOP-SEND MESSAGE TO OPERATOR

The Q5SNDMOP subroutine (refer to figure 8-44) sends a message to the operator. SIL adds the appropriate control characters to the message. If the specified message is longer than 80 characters, SIL truncates the message without returning an error code.

If the operator is not logged on and a message is sent to the operator, the system message buffer becomes full. If the system message buffer is full, the user cannot send a message to the operator. To ensure that the operator sees a message, the user specifies the SAVE parameter on the Q5SNDMOP call so the system saves the message in the save table if the operator is not logged on. The operator can see the saved messages when he accesses the save table. Only one message per task (the one most recently sent) is retained in the save table.

Q5SNDMOP uses the Send Message to Operator system message.

Call Format

CALL Q5SNDMDF('MSG=',msg,optional parameters)

Calling Parameters

'LEN=',len Message length in bytes. If LEN= is omitted, SIL assumes the first character of the message is a delimiter and that the message consists of the second character through the character preceding the next occurrence of the delimiter. SIL sends a maximum of 2000 bytes after removing the delimiters. If the message exceeds that length, SIL truncates it, but does not return an error.

'MSG=',msg Message to be sent. This parameter is required.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 299, 320, 327 through 330.

Figure 8-42. Q5SNDMDF Call Format

Call Format

CALL Q5SNDMJC('MSG=',msg,optional parameters)

Calling Parameters

'LEN=',len Length in bytes of the message to be sent. If LEN= is omitted, SIL assumes the first character of the message is a delimiter and the message consists of the second character through the character preceding the next occurrence of the delimiter. SIL sends a maximum of 2000 bytes after removing the delimiters; if the message exceeds that length, SIL truncates it, but does not return an error.

'MSG=',msg Message to be sent. This parameter is required.

'REJECT' Indicates that if the message would replace an existing message, task execution is suspended until the message can be sent. If REJECT is omitted, this message replaces any existing message. If REJECT is specified, RETURN must be omitted.

'RETURN' Indicates that if the message would replace an existing message waiting for the controller, SIL returns an error code of 323, 324, 332, or 337 (refer to appendix B). If RETURN is specified, REJECT must be omitted.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array in which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 202, 250, 261, 320 through 324, 337.

Figure 8-43. Q5SNDMJC Call Format

Call Format

CALL Q5SNDMOP('MSG=',msg,optional parameters)

Calling Parameters

'LEN=',len Length in bytes of the message to be sent. If LEN= is omitted, SIL assumes the first character of the message is a delimiter and the message consists of the second character through the character preceding the next occurrence of the delimiter. SIL sends a maximum of 80 bytes after removing the delimiters; if the message exceeds that length, SIL truncates it but does not return an error.

'MSG=',msg Message to be sent. This parameter is required.

'RETURN' Indicates that if the message would replace an existing message, SIL returns a status code of 0326. If RETURN is omitted, SIL suspends task execution until it can send the message.

'SAVE' Indicates that SIL should save the message in the save table for later access by the operator. If SAVE is omitted, SIL does not save the message.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array to which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 202, 250, 261, 320, 326.

Figure 8-44. Q5SNDMOP Call Format

Q5SNDSTR-START CONTROLLEE EXECUTION

The Q5SNDSTR subroutine (refer to figure 8-45) starts a controllee task. The user previously initialized the controllee with a Q5INIT call.

Q5SNDSTR uses the Send Message to Controllee system message although it does not send a message to the controllee.

Q5TERM-TERMINATE TASK

The Q5TERM subroutine (refer to figure 8-46) terminates a task and its lower level controllees.

If the user does not specify the RESTART parameter, Q5TERM calls an SIL subroutine to close all task files.

Q5TERM uses the Terminate system message.

Q5TERMCE-DISCONNECT CONTROLLEE

The Q5TERMCE subroutine (refer to figure 8-47) disconnects a previously initialized controllee.

Q5TERMCE uses the Disconnect Controllee system message.

Q5TIME-GET SYSTEM TIME

The Q5TIME subroutine (refer to figure 8-48) gets the system time and date. The user must specify the TIME=, DATE=, JULIAN=, or MASTER= parameter on the call.

Q5TIME uses the Miscellaneous system message.

Call Format

CALL Q5SNDSTR(optional parameters)

Calling Parameters

'DB=',db Descriptor block number identifying the controllee to be started. If DB= is omitted, SIL starts the next lower controllee in the controllee chain.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).
'ERRMSG=',msg 80-byte array to which SIL returns an error message.
'STATUS=',stat Status code. Possible values: 0 through 299, 321.

Figure 8-45. Q5SNDSTR Call Format

Call Format

CALL Q5TERM(optional parameters)

Calling Parameters

'ABORT' Indicates a termination state of #3D (task aborted). ABORT overrides RESTART if both are specified. If ABORT is omitted, the termination state is #3E (normal termination) (refer to Q5GETCTS call description).
'ERROR' Indicates the system return code is 4 (nonfatal errors). ERROR and FATAL are mutually exclusive. If ERROR and FATAL are omitted, the system return code is 0 (no errors).
'FATAL' Indicates the system return code is 8 (fatal errors). ERROR and FATAL are mutually exclusive. If ERROR and FATAL are omitted, the system return code is 0 (no errors).
'RESTART' Indicates that the drop file, scratch files, and output files are to be saved so that the program can be restarted. If RESTART is omitted, the files are not saved. If restarted, the program restarts after the point of termination.
'RESUME=',adr Virtual bit address at which the restarted program should resume execution. The address specified must be in the same subroutine that issued the Q5TERM call.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).
'ERRMSG=',msg 80-byte array in which SIL returns an error message.
'STATUS=',stat Status code. Possible values: 0 through 199, 261.

Figure 8-46. Q5TERM Call Format

Call Format

CALL Q5TERMCE(optional parameters)

Calling Parameters

None.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array in which SIL returns an error message.

'STATUS=',stat Status code. Possible values: 0 through 202, 261, 370.

Figure 8-47. Q5TERMCE Call Format

Call Format

CALL Q5TIME(

'DATE=',date 'JULIAN=',date 'MASTER=',time 'TIME=',time	}	,optional parameters)
--	---	-----------------------

Calling Parameters

None.

Return Parameters

'DATE=',date ASCII character string in the format mm/dd/yy. The variable must begin on a word boundary.

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg 80-byte array in which SIL returns an error message.

'JULIAN=',date Binary Julian date in the following format.

0	48	yy7	ddd9
---	----	-----	------

yy is the year and ddd is the day within the year.

'MASTER=',clk ASCII value of the master clock (Wyman clock). The value is in the format yymmddhhsspppp where yy is the year, mm is the month, dd is the day, hh is the hour, ss is the second, and pppp is the decimal fraction of a second.

'STATUS=',stat Status code. Possible values: 0 through 202, 261.

'TIME=',time ASCII character string in the form hh.mm.ss. The variable must begin on a word boundary.

Figure 8-48. Q5TIME Call Format

Q5VRACC-CHANGE ACCOUNTING RATE

The Q5VRACC subroutine (refer to figure 8-49) changes the accounting rate for the task. Only a public controllee or a controllee whose user directory has the variable rate permit flag set can issue a Q5VRACC call. The site can

set an installation parameter that prevents accounting rate changes.

Q5VRACC uses the Variable Rate Accounting system message.

Call Format

CALL Q5VRACC(optional parameters)

Calling Parameters

VRI=',vri Index (0 through 255) into the variable rate table. If VRI= is omitted, SIL uses index 0.

Return Parameters

ERRLEN=',len Error message length in bytes (integer).

ERRMSG=',msg 80-byte array to which SIL returns an error message.

STATUS=',stat Status code. Possible values: 0 through 299, 470 through 472

Figure 8-49. Q5VRACC Call Format

This section describes SIL input/output (I/O) routines. Section 8 describes non-I/O SIL routines. SIL is a set of subroutines callable by user programs written in FORTRAN, IMPL, or the CYBER 200 Assembler language. Most SIL subroutines format and issue a system message. (System messages are described in volume 2.)

OVERVIEW

By calling SIL routines, the user can read and write files in the formats described in section 3. SIL uses a file information table (FIT) to perform I/O on a file. It uses the system's file index table to access files.

SIL can read or write files on mass storage or magnetic tape. The following are the primary SIL functions and the routines that perform them.

Permanent file access

Q5DEFINE	Defines a permanent file.
Q5ATTACH	Attaches a permanent file.
Q5RETURN	Returns a permanent file or discards a temporary file.
Q5CHANGE	Changes file attributes.
Q5GIVE	Gives file ownership to another user.
Q5PURGE	Purges a permanent file.

Local file and tape file access

Q5REQUEST	Creates or accesses a local or tape file.
Q5CHANGE	Changes file attributes.
Q5RETURN	Returns a local or tape file.

Pool access

Q5PCREAT	Adds a pool to the pool list.
Q5GIVE	Gives a file to a pool.
Q5PGRACC	Grants access to a pool.
Q5PATACH	Attaches a pool of files.
Q5PDTACH	Returns a pool of files.
Q5PREACC	Removes access to a pool.
Q5PURGE	Purges pool file.
Q5PDESTR	Removes a pool from the pool list.
Q5POOLS	Lists the pools.
Q5PUSERL	Lists the users granted access to a pool.

Public file creation

Q5GIVE	Adds a file to the public file list.
--------	--------------------------------------

FIT manipulation

Q5GENFIT	Generates a FIT.
Q5SETFIT	Changes FIT fields.
Q5GETFIT	Retrieves contents of FIT fields.
Q5RETFIT	Returns a FIT.

I/O preparation

Q5OPEN	Opens a file for I/O.
Q5GETFIL	Opens or requests and opens a file.
Q5CLOSE	Closes a file for I/O.

Implicit I/O preparation

Q5MAPIN	Associates a virtual region with a mass storage file.
Q5MAPOUT	Disassociates a virtual region from a mass storage file.

Explicit I/O by physical blocks

Q5READ	Reads from a file.
Q5WRITE	Writes on a file.
Q5CHECK	Checks if I/O operation is complete.

Explicit I/O by logical partitions

Q5GETN	Reads complete partition.
Q5GETP	Reads partial partition.
Q5PUTN	Writes complete partition.
Q5PUTP	Writes partial partition.
Q5ENDPAR	Writes partition delimiter.

File positioning

Q5REWIND	Rewinds file.
Q5SKIP	Skips file partitions forward or backward.

Other functions

Q5REDUCE	Releases allocated mass storage space that is not in use by the file.
Q5ROUTE	Routes a file.
Q5STATUS	Retrieves the current file status.

FILE INFORMATION TABLE (FIT)

SIL coordinates its processing of a file through the FIT it maintains for the file. The FIT format is shown in table 9-1.

To identify a file in an SIL call, the user can specify the file name (logical file name, lfn) or the number SIL assigned to the file's FIT (file logical unit number, flun). The logical file name of a permanent or local file is the file name in its file index entry. File specification by number, rather than by name, is recommended because name specification required SIL to associate the name with a FIT.

SIL must generate a FIT for each file it reads or writes during a task. It discards all FITs at task completion. For existing files, SIL takes FIT information from the file's file index entry as well as from user-supplied parameter values or default values.

If the blocking type (BT) field of a file's file index entry is zero, SIL assumes the file was created before SIL was added to the system; therefore, it enters default values in the SIL fields of the file index entry. (The user can change these values using a Q5CHANGE call.)

The user can explicitly change FIT field values with the Q5SETFIT call and retrieve them with the Q5GETFIT call.

TABLE 9-1. FIT FORMAT

Word	Bits	Field	Contents
0	0-63	lfn	File name; eight ASCII characters.
1	0-2	fo	File organization. 0 Sequential.
1	3-5	bt	Blocking type. 0 Non-SIL file. 2 Character count.
1	6-9	rt	Record type. 0 Control word delimited (W). 1 ANSI fixed length (F). 2 Record mark delimited (R). 7 Undefined (U).
1	10-12	ucs	File access. 0 Not defined (read access granted). 1 Write access. 2 Read access. 3 Read and write access. 4 Write temporary access.
1	13-19	lfp	Logical file position. 0 Within a logical record. 1 Beginning of information. 2 Beginning of file. 4 End of volume. 5 End of file. 8 End of group. 16 End of record. 32 Beginning of volume. 64 End of information.
1	20-21	ofp	File positioning when opened. 0 Not specified (the file is rewound). 1 Rewind the file. 2 Do not rewind the file.
1	22-23	exp	Positioning when end of volume encountered. 0 Unload volume. 1 Rewind volume. 2 Do not rewind volume.
1	24-25	cfp	File positioning when closed. 0 Do not rewind file. 1 Rewind file. 2 Rewind and unload file.
1	26	fnf	Fatal error flag; set when fatal error encountered.
1	27	pef	Parity error flag; set when fatal error encountered.
1	28	rmf	Random mode flag; currently not used.
1	29	srf	Flag indicating whether control is returned immediately to the caller after issuance of an I/O request; if set, control is not returned until the I/O request is complete.

TABLE 9-1. FIT FORMAT (Contd)

Word	Bits	Field	Contents
1	30-31	ocs	Open or closed file status. 0 Never opened. 1 Opened for explicit I/O. 2 Closed. 3 Opened for implicit I/O.
1	32	cnf	Connected file flag; if set, the file is connected to a terminal.
	33	wpf	Write flag; if set, the last operation was a write operation.
	34	bsf	Buffer specified flag; if set, Q5OPEN, Q5GETFIT, or Q5SETFIT specified a buffer.
1	35	peof	End of tape flag; if set, the file is positioned at the end of the tape.
1	36	cef	Compression/expansion flag; if set, blank compression and expansion is performed on the file.
1	37-47		Reserved.
1	48-55	pc	Padding character; one ASCII character.
1	56-63	rmk	Record-mark character; one ASCII character.
2	0-15	llop	Last logical operation on the file requested by the user. 1 Q5DEFINE 13 Q5PUTN 2 Q5MAPIN 14 Q5PUTP 3 Q5MAPOUT 15 Q5READ 4 Q5OPEN 16 Q5WRITE 5 Q5CLOSE 17 Q5REWIND 6 Q5PURGE 18 Q5SKIP 7 Q5RETURN 19 Q5CHANGE 8 Q5REQUEST 20 Q5GIVE 9 Q5CHECK 21 Q5REDUCE 10 Q5ENDPAR 22 Q5ROUTE 11 Q5GETN 23 Q5STATUS 12 Q5GETP
2	16-31	lvsc	Last virtual system function code issued for the file.
2	32-47	lvso	Last virtual system suboperation code issued for the file.
2	48-63		Reserved.
3	0-7	ioc	Number of the system input/output connector (IOC) used by the file.
3	8-15	buf1l	Length in 512-word blocks of buffer one.
3	16-63	buf1	Address of buffer one.
4	0-5		Reserved.
4	6-7	tpm	Tape mode. 0 BCD, 7-track, even parity. 1 Binary, 7- or 9-track, odd parity. 2 6-bit ASCII, 7-track, even parity. 3 8-bit ASCII, 7 or 9-track, even parity.

TABLE 9-1. FIT FORMAT (Contd)

Word	Bits	Field	Contents
4	8-15	buf12	Length in 512-word blocks of buffer two.
4	16-63	buf2	Address of buffer two.
5	0-15	ws1	Length in bytes of the working storage area.
5	16-63	wsa	Address of the working storage area.
6	0-31	es	Last SIL status code.
6	32-47	ect	Warning error count.
6	48-63	erl	Warning error limit.
7	0-63		Reserved.
8	0-31	rc	Record count; number of the last full record read or written.
8	32-63	bn	Ordinal of current block.
9	0-4	rsn	Request serial number of last Q5READ or Q5WRITE call. It also identifies the word within the FIT (st1 through st6) containing the call response.
9	5-7	cltyp	Close type (as determined by the Q5OPEN call). 0 Nonprivileged. 1 Privileged. 2 USER1.
9	8-15	unit	Logical unit number of the device on which the file resides as set by the system.
9	16-39	mnr	Minimum record length.
9	40-63	mnr	Maximum record length or fixed record length.
10	0-7		Reserved.
10	8-15	try	Error recovery for tape files. 0 Attempt error recovery; discard noise records. 1 Do not attempt error recovery; discard noise records. 2 Attempt error recovery; use noise records. 3 Do not attempt error recovery; use noise records.
10	16-39	rl	Current record length.
10	40-63	ptl	Current partial transfer length.
11	0-1	cbn	Current buffer being used. 0 Both buffers free. 1 Buffer one in use. 2 Buffer two in use.
11	2-15		Reserved.
11	16-63	cbo	Current byte position within the buffer.
12	0-2	lt	Label type. 0 ANSI standard labels. 1 No labels.

TABLE 9-1. FIT FORMAT (Contd)

Word	Bits	Field	Contents
12	3-5		Reserved.
12	6-8	den	Tape density. 0 200 bpi 1 556 bpi 2 800 bpi 3 1600 bpi
12	9-16	vn	Tape file version number for HDR1 label.
12	17-27	rp	Tape file retention period for HDR1 label.
12	28-39	dt	Device type. 0 Mass storage. 7 7-track magnetic tape. 9 9-track magnetic tape.
12	40-63	pno	Tape file position within multifile set. (0 is load point.)
13	0-63		Reserved.
14	0-63	fid	Tape file identifier for HDR1 label; 17 ASCII characters.
15	0-63	fid	
16	0-7	fid2	
16	8-55	stid	Tape file set identifier for HDR1 label; six ASCII characters.
16	56-63	acst	Accessibility character for HDR1 label.
17	0-15	gn	Tape file generation number for HDR1 label.
17	16-63	vsn	Volume serial number for VOL1 label.
18	0-15	lvsn	Length in words of the user-specified VSN list.
18	16-63	bvsn	Address of user-specified VSN list.
19-24		st1-st6	Status words used by an internal routine for physical I/O operations.
25	0-15	sbn1	Starting block number in buffer one.
25	16-63	eod1	Byte offset to end of data in buffer one.
26	0-15	sbn2	Starting block number in buffer two.
26	16-63	eod2	Byte offset to end of data in buffer two.
27	0-63	owner	First eight characters of owner identification in VOL1 label.
28	0-47	owner2	Last six characters of owner identification in VOL1 label.
28	48-63		Reserved.

TAPE LABEL PROCESSING

SIL supports level 2 tape processing as defined in the American National Standard for Magnetic Tape Labels and File Structure for Information Interchange, X3.27-1978. The supported tape file sets are:

- Single file, single volume
- Multifile, single volume
- Single file, multivolume
- Multifile, multivolume

Each labeled volume begins with a VOL1 label and ends with an EOVI label. Each file begins with a HDR1 label and ends with an EOF1 label. The label formats are in table 9-2.

To access a tape file, the user must specify the device type on the Q5RQUEST call as either 9-track tape or 7-track tape. The user must also specify read access or write access.

To access an unlabeled tape, the user specifies the NOLABEL parameter. To access a labeled tape, the user omits the NOLABEL parameter.

To write the VOL1 label on a tape, the tape must be unlabeled or the operator must enter a command to write the VOL1 label. The user specifies the NEWLAB parameter on the Q5RQUEST call. SIL records the volume serial number (VSN) in the VOL1 label.

To write the HDR1 label, the user must specify the ACS= parameter on the Q5OPEN call. If the user requests new labels, the Q5OPEN call checks whether the expiration date in the existing HDR1 label has passed. It also checks whether the accessibility character in the existing HDR1 label is blank. If the character is not blank, it checks whether the user specified the matching accessibility character. After performing these validation checks, it writes the HDR1 label using the label field values specified on the call or in the file's FIT or default values.

SIL writes the EOVI label when it senses the end of the tape and the EOF1 label when the file is closed.

To read a tape, the user specifies read access and omits the NEWLAB parameter. When the user requests a labeled tape using the Q5RQUEST call, SIL ensures that the operator mounted the correct volume by comparing the VSN specified on the call with the VSN specified in the VOL1 label. When the user opens the labeled tape file using the Q5OPEN call, SIL checks the accessibility character. It also checks that the label field values specified on the call match the values in the HDR1 label.

To read or write a multifile set, the user must specify the set identifier with the STID= parameter.

SIL I/O CALLS

This section contains a figure for each SIL routine. The figure contains a call format specifying the required parameters followed by parameter descriptions. The parameter descriptions are divided between calling parameters and return parameters. A calling parameter specifies a value used by the SIL routine. A return parameter specifies the name of the variable in which SIL returns a value.

Parameter keywords are listed as FORTRAN literals. Options are listed as lowercase variable names. The available mnemonic values for calling parameter options are listed as FORTRAN literals.

Q5ATTACH-ATTACH PERMANENT FILE

The user calls the Q5ATTACH routine (refer to figure 9-1) to access one or all of his permanent files.

If, while attaching all of the user's permanent files, Q5ATTACH encounters an error preventing it from attaching a file, it records the error and continues attaching files; therefore, the status code returned is that of the last error encountered.

TABLE 9-2. ANSI LABEL FORMATS

Label	Character Position	Field	Name	Length	Contents	Default Written
Volume header	1 to 3	1	Label Identifier	3	VOL	VOL
	4	2	Label Number	1	1	1
	5 to 10	3	Volume Serial Number	6	Any characters	
	11	4	Accessibility	1	Space	Space
	12 to 31	5	Reserved	20	Spaces	Spaces
	32 to 37	6	Reserved	6	Spaces	Spaces
	38 to 51	7	Owner ID	14	Any characters	Spaces
	52 to 79	8	Reserved	28	Spaces	Spaces
	80	9	Label Standard Level	1	1	1
First file header	1 to 3	1	Label Identifier	3	HDR	HDR
	4	2	Label Number	1	1	1
	5 to 21	3	File Identifier	17	Any characters	Spaces
	22 to 27	4	Set Identification	6	Any characters	Volume serial number of first reel of the set
	28 to 31	5	File Section Number	4	Four digits indicating number of volume in file	0001
	32 to 35	6	File Sequence Number	4	Four digits indicating number of file in multifile set	0001
	36 to 39	7	Generation Number	4	(Not used by the operating system)	Spaces
	40, 41	8	Generation Version Number	2	Two digits indicating the edition of the file	00
	42 to 47	9	Creation Date	6	Space followed by two digits for year, three digits for day	Current date is used
	48 to 53	10	Expiration Date	6	Same as field 9	Same as field 9
	54	11	Accessibility	1	Any characters	Space
	55 to 60	12	Block Count	6	Zeros	Zeros
	61 to 73	13	System Code	13	Any characters	Spaces
	74 to 80	14	Reserved	7	Spaces	Spaces
First end-of-file	1 to 3	1	Label Identifier	3	EOF	EOF
	4	2	Label Number	1	1	1
	5 to 54	3 to 11	Same as corresponding HDR1 label fields			

TABLE 9-2. ANSI LABEL FORMATS (Contd)

Label	Character Position	Field	Name	Length	Contents	Default Written
First end-of-file (contd)	55 to 60	12	Block Count	6	Six digits indicating number of data blocks since last HDR label group	
	61 to 80	13, 14	Same as corresponding HDR1 label fields			
First end-of-volume	1 to 3	1	Label Identifier	3	EOV	EOV
	4	2	Label Number	1	1	1
All other fields are identical to EOF1 label.						

Call Format

CALL Q5ATTACH ({ 'LFN=', lfn } , optional parameters)

Calling Parameters

'LFN=', lfn Name of an unattached permanent file. LFN=, lfn must be specified if * is omitted.

'*' Indicates that SIL attaches all unattached, permanent files belonging to the user. * must be specified if LFN=, lfn is omitted.

Return Parameters

'ERRLEN=', len Error message length in bytes.

'ERRMSG=', msg Error message. The variable msg must be 80 bytes long.

'STATUS=', stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-1. Q5ATTACH Call Format

Q5CHANGE-CHANGE FILE ATTRIBUTES

The user calls the Q5CHANGE routine (refer to figure 9-2) to change file attributes. Q5CHANGE records the specified changes in the file's file index entry and in its

FIT. Because the changes are recorded in the file index, the changes are permanent for a permanent file and carry over to the next time the file is attached.

Unless the user is privileged, he can change the attributes of his private files only.

Call Format

CALL Q5CHANGE($\left\{ \begin{array}{l} \text{'LFN=,lfn} \\ \text{'FLUN=,rflun} \end{array} \right\}$, optional parameters)

Calling Parameters

'LFN=,lfn	File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.
'FLUN=,rflun	Number SIL assigned to the file and its FIT. FLUN=,rflun must be specified if LFN=,lfn is omitted.
'ACS=,acs	File access permission. If ACS=,acs is omitted, SIL does not change the file access permission. <div><div>'N'</div>No read or write access. <div>'R'</div>Read access. <div>'W'</div>Write access. <div>'RW'</div>Read and write access. <div>'T'</div>Write temporary access.</div>
'BT=,bt	Blocking type. If BT=,bt is omitted, SIL does not change blocking type. <div>'C'</div> Fixed character count.
'DFLEN=,dfl	Drop file length in 512-word blocks. If DFLEN=,dfl is omitted, SIL does not change the drop file length.
'FC=,fc	File category. If FC=,fc is omitted, SIL does not change file category. <div>'B'</div> Batch file. <div>'U'</div> User file.
'MNR=,mnr	Minimum record length in bytes. For record types other than F, SIL checks that a record is not shorter than this value. SIL does not use this value when writing F format records. If MNR=,mnr is omitted, SIL does not change the minimum record length.
'MXR=,mxr	Maximum record length. For F format records, mxr is the fixed record length. For other record formats, SIL checks that the record length does not exceed this value. If MXR=,mxr is omitted, SIL does not change the maximum record length.
'NFNAME=,nf	New file name. A Q5CHANGE call cannot change both the file name and the retention period (RP=,rp). If NFNAME=,nf is omitted, SIL does not change the file name.
'PC=,pc	Padding character used to fill the working storage area. If PC=,pc is omitted, SIL does not change the padding character.
'RP=,rp	Retention period in days. A Q5CHANGE call cannot change both the retention period and the file name (NFNAME=,nf). If RP=,rp is omitted, SIL does not change the retention period.
'RT=,rt	Record format. If RT=,rt is omitted, SIL does not change the record format. <div>'F'</div> ANSI fixed length. <div>'R'</div> Record mark delimited. <div>'U'</div> Undefined structure. <div>'W'</div> Control word.

Figure 9-2. Q5CHANGE Call Format (Sheet 1 of 2)

'RMK=',rmk	Record mark character. If RMK=,rmk is omitted, SIL does not change the record mark character.
'SFO=',sfo	File organization. If SFO=,sfo is omitted, SIL does not change the file organization.
	'S' Sequential organization.
'TYPE=',typ	File type. If TYPE=,typ is omitted, SIL does not change the file type.
	'PD' Physical data file.
	'VC' Virtual data file.

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-2. Q5CHANGE Call Format (Sheet 2 of 2)

Q5CHECK-CHECK I/O REQUEST STATUS

The user calls the Q5CHECK routine (refer to figure 9-3) to determine the status of a Q5READ or Q5WRITE request. If more than one Q5READ or Q5WRITE request is outstanding, the user can identify the request by its request ordinal. The RSN parameter on the Q5READ or Q5WRITE call returns the request ordinal.

After determining the request status by examining the appropriate fields in the FIT, SIL returns control immediately to the user by default. If the user specifies the WAIT parameter on the Q5CHECK call; however, SIL suspends program execution until the I/O request has completed.

If the I/O request is a tape read, Q5CHECK can return a count of the bits read in excess of the byte count returned by the RL parameter. SIL is most likely to return a nonzero excess bit count when reading a tape written on a system other than a CYBER 200. This could be a result of word size incompatibility between systems.

Q5CLOSE-CLOSE FILE

The user calls the Q5CLOSE routine (refer to figure 9-4) to close one or all open files. Closing a file severs the I/O connection between the file and the task. Normal job termination closes all open files. The user should close files after completing I/O in case the job terminates abnormally.

If the last operation on the file was a write operation, Q5CLOSE writes any data remaining in the output buffers and writes an end-of-file indicator.

If the file was opened for explicit I/O with write access, the modified pages are rewritten on mass storage. If the file does not have write access, the modified pages are discarded.

If the file is an output file with no other tasks accessing it, the file is output after it is closed.

If the file is on a labeled tape and the last operation was a write operation, Q5CLOSE writes the EOF1 label. The user can specify that Q5CLOSE rewind or unload the tape. If, while closing a tape file, Q5CHECK encounters the EOF1 label or the end of tape indicator, it rewinds and unloads the volume by default; requests the next volume if the file is a multivolume set; and reads its VOL1 and HDR1 labels. To write a multifile set, the user must close the tape file and then reopen it to write the next file.

If, while closing all open files, Q5CLOSE encounters an error preventing it from closing a file, it records an error status code and continues closing files; therefore, the status code returned is that of the last error encountered.

The system limits to 110 the number of FITs that can be concurrently associated with a job. The user can return a FIT by specifying the RETFIT parameter on the Q5CLOSE call or by issuing a Q5RETFIT call. After SIL returns a file's FIT, it cannot reopen the file until a new FIT is generated.

Call Format

CALL Q5CHECK({ 'LFN=',lfn
'FLUN=',rflun } ,IOSTAT=',io,optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=',lfn must be specified if FLUN=',rflun is omitted.

'FLUN=',rflun Number SIL assigned to the file. FLUN=',rflun must be specified if LFN=',lfn is omitted.

'RSN=',rsn Request number of the request to be checked. SIL assigns request numbers to identify concurrent I/O request. Q5READ and Q5WRITE return an RSN value. If RSN=',rsn is omitted, SIL checks the last issued I/O request.

'WAIT' Indicator that SIL should wait for completion of the I/O request (specified by RSN) before returning control to the caller.

Return Parameters

'ERRLEN=',len Error message length in bytes.

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'IOSTAT=',sts Status of the I/O request. This parameter is required. SIL can return the following ASCII values.

COM	I/O request completed without errors.
EOF	I/O request completed; end of file encountered.
ERR	I/O request completed with errors.
PEN	I/O request pending; errors encountered.

'RL=',rl Record length (in bytes) actually transferred.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Return Parameter for Tapes Only

'XBC=',xbc Excess bits read. If SIL returns a nonzero value after reading a tape written by a CYBER 200 system, the byte count that Q5READ returns as the record length (LEN=',len) should be decremented by one.

Figure 9-3. Q5CHECK Call Format

Call Format

CALL Q5CLOSE($\left\{ \begin{array}{l} \text{'LFN=,lfn} \\ \text{'FLUN=,rflun} \\ \text{'*'} \end{array} \right\}$, optional parameters)

Calling Parameters

'LFN=,lfn Name of the open file SIL closes. LFN=,lfn must be specified if * and FLUN=,rflun are omitted.

'FLUN=,rflun Number SIL assigned to the open file. FLUN=,rflun must be specified if * and LFN=,lfn are omitted.

'*' Indicates that SIL should close all open files. * must be specified if LFN=,lfn and FLUN=,rflun are omitted.

'CFP=,cfp File positioning. If CFP=,cfp is omitted, SIL does not rewind the file.

 'N' Do not rewind file. Tape files are positioned past the EOF1 label.

 'R' Rewind file.

 'U' Unload file.

'FC=,cat Indicates that when SIL closes the file, it should change the file category of the file.

 'D' Drop file.

'RETFIT' Indicates that SIL should discard the FIT for the file. If RETFIT is omitted, SIL can reopen the file with this FIT.

Return Parameters

'ERRLEN=,len Error message length in bytes (integer).

'ERRMSG=,msg Error message. The variable msg must be 80 bytes long.

'STATUS=,stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-4. Q5CLOSE Call Format

Q5DEFINE-DEFINE PERMANENT FILE

The user calls the Q5DEFINE routine (refer to figure 9-5) to create a permanent file. The new permanent file can be a newly created file or an existing local file. The user can specify file attributes for a newly created file, but not for an existing file.

The action Q5DEFINE performs depends on the file name or number specified.

<u>If Q5DEFINE specifies:</u>	<u>Result</u>
-------------------------------	---------------

A permanent file	Error status returned.
------------------	------------------------

A local file	The local file becomes a permanent file attached to the job. The Q5DEFINE call does not change the open or closed status of the file.
--------------	---

<u>If Q5DEFINE specifies:</u>	<u>Result</u>
-------------------------------	---------------

A non-existing file	A new permanent file is created, closed and attached to the job.
---------------------	--

The new permanent file is not attached to other jobs running under the same user number. If necessary, Q5DEFINE generates a FIT for the new file. If an error occurs during processing, Q5DEFINE does not destroy the FIT.

Privileged Q5DEFINE calls use only the following parameters: LFN= or FLUN=, EXT=, LEN=, NOSEG, FITE=, and FILEL=. Any other parameters specified are ignored. A privileged Q5DEFINE call does not use the values stored in an existing FIT. If a FIT does not exist for the file, it generates one using default values for each field; however, the FIT values used are those the user specifies via a privileged Q5OPEN call.

Call Format

CALL Q5DEFINE($\left\{ \begin{array}{l} \text{'LFN=,lfn} \\ \text{'FLUN=,rflun} \end{array} \right\}$,optional parameters)

Calling Parameters

'LFN=,lfn Name of the new permanent file. It can be the name of a local file that is to become a permanent file. LFN=,lfn must be specified if FLUN=,rflun is omitted.

'FLUN=,rflun Number SIL assigned to the FIT or local file. FLUN=,rflun must be specified if LFN=,lfn is omitted.

Calling Parameters Used Only When Creating a File

'ACS=,acs File access permission. If ACS=,acs is omitted, SIL allows read and write access.

 'W' Write access.
 'R' Read access.
 'RW' Read and write access.

'BT=,bt Blocking type. If BT=,bt is omitted, the file has fixed character count blocking.

 'C' Fixed character count.

'EXT=,ext File extensibility indicator. If EXT=,ext is omitted, the file is extendible.

 'Y' The file is extendible.
 'N' The file is not extendible.

'FC=,fc File category. If FC=,fc is omitted, the file is a user file.

 'B' Batch file (batch processor controllee).
 'U' User file.

'LEN=,fl File length in 512-word blocks. If LEN=,fl is omitted, the file is eight 512-word blocks.

'MNR=,mnr Minimum record length in bytes. For record formats other than F, SIL checks that the record is not shorter than this value. SIL does not use this parameter when writing F format records. If MNR=,mnr is omitted, SIL assumes the minimum record length is one byte.

'MXR=,mxr Maximum record length in bytes. For F records, mxr is the fixed record length. For other records, SIL checks that the record is not longer than this value. If MXR=,mxr is omitted, SIL assumes maximum record length is the default set by an installation parameter.

'NOSEG' Indicator that file must be contiguous (not written in segments). If NOSEG is omitted, SIL can segment the file.

'PN=,pn Six-character identifier of the disk pack on which SIL creates the file. If PN=,pn is omitted, the system assigns mass storage space for the file.

'PC=,pc Padding character used to fill the working storage area. If PC=,pc is omitted, SIL pads with blanks.

'RMK=,rmk Record delimiting character for R format records. If RMK=,rmk is omitted, SIL uses the installation-specified character (usually ASCII US, #1F code).

'RT=,rt Record format. If RT=,rt is omitted, SIL assumes the default set by an installation parameter.

 'F' ANSI fixed length.
 'R' Record mark delimited.
 'U' Undefined.
 'W' Control word.

'SFO=,fo File organization. If SFO=,fo is omitted, SIL assumes sequential organization.

 'S' Sequential organization.

Figure 9-5. Q5DEFINE Call Format (Sheet 1 of 2)

'SLEV=',sl Security level, (1 through 255, but less than or equal to that of the caller). If SLEV=',sl is omitted, SIL sets the file security level equal to that of the caller.

'TYPE=',typ File type. If TYPE=',typ is omitted, SIL assumes the file is a physical data file.

 'PD' Physical data file.

 'VC' Virtual code file.

Calling Parameters for Privileged Users Only

'FITE=',array Name of array containing a copy of a file index entry. SIL uses the copy to initialize the file index entry for the file. If FITE=',array is omitted, SIL generates the file index entry.

'FITE=',alen Length (in words) of the array named by the FITE=',array parameter. The system checks that this length is the length of a file index entry. If FITE=',alen is omitted, SIL generates the file index entry.

Return Parameters

'CONT=',con Initial contiguity of the mass storage file. SIL can return the following ASCII values:

 Y The file space is contiguous.

 N The file space is segmented.

'DA=',da Action performed by the Q5DEFINE call. SIL can return the following ASCII values:

 N New permanent file created.

 Y Local file made permanent.

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'MLEN=',max Maximum length of the file in 512-word blocks (refer to File Space Allocation in section 2).

'RFLUN=',rflun Number SIL assigned to the file.

'RPN=',pn Six-character identifier of the disk pack on which the file resides.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

'UNIT=',dn Logical device number for the unit on which the file resides.

Return Parameters for Privileged Users Only

'FSTO=',fst File segment table ordinal. If the attempt to create a file failed, SIL does not return an ordinal value.

Figure 9-5. Q5DEFINE Call Format (Sheet 2 of 2)

Q5ENDPAR-WRITE PARTITION DELIMITER

The user calls the Q5ENDPAR routine (refer to figure 9-6) to write a partition delimiter on the file. Q5ENDPAR can write delimiters to mark the end of a record, a group, or a file. It can also write a tape mark on an unlabeled tape.

The Q5PUTN routine automatically writes partition delimiters. When the user issues Q5PUTN calls to write a file, Q5ENDPAR calls are not necessary.

U or F format files cannot contain record or group delimiters. They can contain only one end-of-file delimiter; therefore, the user should not issue Q5ENDPAR calls for those files.

Q5GENFIT-GENERATE FIT

The user calls the Q5GENFIT routine (refer to figure 9-7) to generate and initialize a file information table (FIT) for the specified file name. Q5GENFIT uses parameter specifications and default values to initialize the fields in the FIT.

SIL requires a FIT for each file it reads or writes. The Q5ATTACH, Q5DEFINE, and Q5RQUEST calls generate FITs for their files. The system discards those FITs at completion of the task; however, the local and attached permanent files remain attached to the job. The user can generate a new FIT for a file attached by a previous task with a Q5GENFIT call or a Q5OPEN call.

Call Format

CALL Q5ENDPAR({ 'LFN=',lfn }
{ 'FLUN=',rflun } ,optional parameters)

Calling Parameters

'LFN=',lfn	File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.
'FLUN=',rflun	Number SIL assigned to the file FLUN=,rflun must be specified if LFN=,lfn is omitted.
'PART=',part	File partition delimiter. If PART=,part is omitted, SIL writes a record delimiter.
	'R' Record delimiter.
	'G' Group delimiter.
	'F' File delimiter.
	'T' Tape mark.

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg is 80 bytes long.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-6. Q5ENDPAR Call Format

Call Format

CALL Q5GENFIT('LFN=',lfn, optional parameters)

Calling Parameters

'LFN=',lfn	Name of the file for which SIL generates a FIT. This parameter is required.
'ACS=',acs	File access permission. If ACS=,acs is omitted, SIL allows read and write access. 'W' Write access. 'R' Read access. 'RW' Read and write access.
'BT=',bt	Blocking type. If BT=,bt is omitted, the file has fixed character count blocking. 'C' Fixed character count.
'BUFL1=',bl1	Buffer one length n 512-word blocks. If BUFL1=,bl1 is omitted, SIL assumes a buffer length of three blocks.
'BUFL2=',bl2	Buffer two length in 512-word blocks. If BUFL2=,bl2 is omitted, SIL assumes a buffer length of three blocks.
'BUF1=',b1	Array to be used as data buffer one. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. This parameter is required if SIL is to read or write the file.
'BUF2=',b2	Array to be used as data buffer two. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. Data buffer two is not required.
'CFP=',cfp	File positioning when the file is closed. If CFP=,cfp is omitted, SIL does not rewind the file. 'N' Do not rewind file. Tape files are positioned past the EOF1 label. 'R' Rewind file. 'U' Unload file.
'ERL=',erl	Maximum number of SIL warning errors allowed for the file before SIL aborts the task. If a zero limit is specified or ERL=,erl is omitted, SIL allows an unlimited number of warning errors.
'MNR=',mnr	Minimum record length in bytes. For record types other than F, SIL checks that a record is not shorter than this value. SIL does not use this value when writing F-type records. If MNR=,mnr is omitted, SIL assumes the minimum record length is one byte.
'MXR=',mxr	Maximum record length in bytes. For F-type records, mxr is the fixed record length. For other records types, SIL checks that the record is not longer than this value. If MXR=,mxr is omitted, SIL assumes the maximum record length is the default set by an installation parameter.
'OFP=',ofp	File positioning when the file is opened. If OFP=,ofp is omitted, SIL rewinds the file. 'R' Rewind the file. 'N' Do not rewind the file.
'PC=',pc	Padding character used to fill the working storage area. If PC=,pc is omitted, SIL pads with blanks.
'RMK=',rmk	Record delimiting character for R format records. If RMK=,rmk is omitted, SIL uses the installation-specified character (usually ASCII US, #1F code).
'RT=',rt	Record format. If RT=,rt is omitted, SIL assumes the default set by an installation parameter. 'F' ANSI fixed length. 'R' Record mark delimited. 'U' Undefined. 'W' Control word.
'SFO=',fo	File organization. If SFO=,fo is omitted, SIL assumes sequential organization. 'S' Sequential organization.

Figure 9-7. Q5GENFIT Call Format (Sheet 1 of 2)

'SRF=',srf	Indicate that SIL must complete an I/O request before returning control to the caller. If SRF=,srf is omitted, SIL can return control to the caller before completing the read or write.				
	<table border="0"> <tr> <td style="padding-right: 20px;">'Y'</td> <td>Suppress overlapped I/O.</td> </tr> <tr> <td>'N'</td> <td>Allow overlapped I/O.</td> </tr> </table>	'Y'	Suppress overlapped I/O.	'N'	Allow overlapped I/O.
'Y'	Suppress overlapped I/O.				
'N'	Allow overlapped I/O.				

'WSA=',wsa	Working storage area used by get and put calls.
------------	---

'WSL=',wsl	Length (in bytes) of the working storage area.
------------	--

Calling Parameters for Tape Files Only

'DEN=',den	Tape density. If DEN=,den is omitted, SIL assumes 1600 cpi.
------------	---

'200'	200 bpi (7-track tape)
'556'	556 bpi (7-track tape)
'800'	800 bpi or cpi (7- or 9-track tape)
'1600'	1600 cpi (9-track tape)

'LT=',lt	Label type. If LT=,lt is omitted, SIL assumes ANSI standard labels.
----------	---

'S'	ANSI standard labeled tape.
'U'	Unlabeled tape.

'STID=',st	Six-character set identifier. The user must specify a set identifier for a file in a multifile set.
------------	---

'TPM=',tpm	Tape mode. If TPM=,tpm is omitted, SIL reads or writes tape data in 8-bit ASCII character code.
------------	---

'BCD'	Binary coded decimal for 7-track tapes; even parity.
'BIN'	Unformatted binary for 7- or 9-track tapes; odd parity.
'AS6'	6-bit ASCII code for 7-track tape; even parity
'ASC'	8-bit ASCII code for 7- or 9-track tape; odd parity.

'VSN=',vsn	Six-character volume serial number (VSN) of the requested tape. If VSN=,vsn is omitted or the specified VSN is zero, the operator assigns a tape to the file.
------------	---

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
---------------	--

'ERRMSG=',msg	Error message. The variable msg is 80 bytes long.
---------------	---

'RFLUN=',rflun	Number SIL assigned to the file.
----------------	----------------------------------

'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.
----------------	---

Figure 9-7. Q5GENFIT Call Format (Sheet 2 of 2)

Q5GETFIL-OPEN OR CREATE AND OPEN FILE

The Q5GETFIL routine (refer to figure 9-8) either opens an existing file or requests and opens a new local file. The action taken depends on the file specified and the presence or omission of the RETURN parameter on the call. The possible actions are summarized as follows.

			If the specified file is:	RETURN omitted	RETURN specified
			An attached permanent file	Opens the existing permanent file	Opens the existing permanent file.
			An attached pool file	Opens the existing pool file.	Requests and opens a new local file.
			A public file	Requests and opens a new local file.	Requests and opens a new local file.
			If the user specifies the IMP parameter on the Q5GETFIL call to open the file for implicit I/O, he must also call Q5MAPIN to map in the file.		
If the specified file is:	RETURN omitted	RETURN specified			
Not assigned to job	Requests and opens a new local file.	Requests and opens a new local file.			
A local file	Opens the existing local file.	Returns the existing local file and requests and opens a new local file.			

Call Format

CALL Q5GETFIL('LFN=',lfn, optional parameters)

Calling Parameters

'ACS=',acs	File access permission. If ACS=,acs is omitted, and Q5GETFIL requested the file, read and write access is permitted. If ACS=,acs is omitted and the file already exists, the access specified in the file's FIT is permitted.
	'RW' Read and write access. 'R' Read access. 'W' Write access. 'T' Write-temporary access.
'ALEN=',alen	Length (in words) of the array specified on the MAPS= parameter. If ALEN=,alen is omitted, the file maps are not copied.
'BT=',bt	Blocking type. If BT=,bt is omitted, fixed character count blocking is assumed.
	'C' Fixed character count.
'BUFL1=',bl1	Buffer one length in 512-word blocks. If BUFL1=,bl1 is omitted, buffer length of 3 blocks is assumed.
'BUFL2=',bl2	Buffer two length in 512-word blocks. If BUFL2=,bl2 is omitted, buffer length of 3 blocks is assumed.
'BUF1=',b1	Array to be used as data buffer one. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. This parameter is required if SIL is to read or write the file.
'BUF2=',b2	Array to be used as data buffer two. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. Data buffer two is not required.
'DC=',dc	Disposition code. If DC=,dc is omitted and Q5GETFIL requests the file, the system default disposition code is used; if DC=,dc is omitted and the file already exists, its disposition code is not changed.
	'IN' Batch job input to access station. 'LR' Print on 580-12 printer on the front-end processor. 'LS' Print on 580-16 printer on the front-end processor. 'LT' Print on 580-20 printer on the front-end processor. 'PF' Store as a permanent file. 'PR' Print on any available line printer. 'PU' Punch file. 'P1' Print on 501 printer on the front-end processor. 'P2' Print on 512 printer on the front-end processor. 'SC' Discard file at end of task.

Figure 9-8. Q5GETFIL Call Format (Sheet 1 of 4)

'DT=',dt	Device type on which the file is to reside. DT=,dt is ignored if Q5GETFIL opens an existing file. If DT=,dt is omitted and Q5GETFIL requests the file, the file resides on mass storage; if the file already exists, its residence is not changed. 'MS' Mass storage. 'MT' 7-track magnetic tape. 'NT' 9-track magnetic tape.
'FC=',fc	File category. FC=,fc is ignored if the file already exists. If FC=,fc is omitted and Q5GETFIL requests the file, it requests a user file. 'B' Batch file. 'U' User file.
'IC=',ic	File format. If IC=,ic is omitted, and Q5GETFIL requests the file, the file format is the system default; if the file already exists, its file format is not changed. 'AS' 8-bit ASCII code; ANSI carriage control if print file. 'BI' Binary. 'PA' 8-bit ASCII code; ASCII carriage control if print file.
'IMP'	Indicates the file is to be opened for implicit I/O. The record type in the file's FIT is changed to undefined; the record type in the file's FILEI entry is not changed. If IMP is omitted, the file is opened for explicit I/O.
'LEN=',len	File length in 512-word blocks. LEN=,len is ignored if the file already exists. If LEN=,len is omitted and Q5GETFIL requests the file, its length is eight blocks.
'LFN=',lfn	File name. LFN=,lfn is required.
'MNR=',mnr	Minimum record length in bytes. If MNR=,mnr is omitted and Q5GETFIL requests the file, the minimum record length is one byte. If MNR=,mnr is omitted and file already exists, its minimum record length is not changed.
'MXR=',mxr	Maximum record length in bytes. If MXR=,mxr is omitted and Q5GETFIL requests the file, no maximum record length is set. If MXR=,mxr is omitted and the file already exists, its maximum record length is not changed.
'NOCOMP'	Indicates that blank compression and expansion should not be performed on the file. If NOCOMP is omitted, blank compression and expansion are performed.
'NOEXT'	Indicates that the file cannot be extended. If NOEXT is omitted, the file can be extended.
'NOSEG'	Indicates that the file cannot be segmented. NOSEG is ignored if the file already exists and is not returned. If NOSEG is omitted and Q5GETFIL requests the file, the file can be segmented.
'PC=',pc	ASCII padding character. If PC=,pc is omitted, the blank character is used.
'PN=',pn	Six-character identifier of the disk pack on which the file is created. PN=,pn is ignored if the file already exists and is not returned. If PN=,pn is omitted and Q5GETFIL requests the file, the system determines the file residence.
'RETURN'	Indicates the file is to be returned if its is an existing local file. If RETURN is omitted, the existing local file is opened. Refer to the call description.
'RMK=',rmk	ASCII record delimiting character for R format records. If RMK=,rmk is omitted, the installation-defined record delimiter is used.
'RT=',rt	Record type. Rt=,rt is ignored if the file already exists and is not returned. If RT,rt is omitted, SIL assumes the default set by an installation parameter. 'F' ANSI fixed length. 'R' Record mark delimited. 'U' Undefined. 'W' Control word.
'SFO=',fo	File organization. If SFO=,sfo is omitted, sequential organization is assumed. 'S' Sequential organization.

Figure 9-8. Q5GETFIL Call Format (Sheet 2 of 4)

'SLEV=',sl	Security level (1 through 255 but less than equal to that of the caller). SLEV,sl is ignored if the file already exists and is not returned. If SLEV,sl is omitted and Q5GETFIL requests the file, its security level is that of the caller.
'TYPE=',typ	File type. 'TYPE=',typ is ignored if the file already exists and is not returned. If TYPE=,typ is omitted and Q5GETFIL requests the file, the file is a physical data file.
'WSA=',wsa	Working storage area used by get and put I/O calls.
'WSL=',wsl	Length (in bytes) of the working storage area.

Calling Parameters for Privileged Users Only

'FITE=',array	Name of the array in which a copy of the file index table for the file is returned. The user must enter the user number or pool name in the first word of the array and the file name in the second word (refer to the file index entry format in volume 2).
'FITLE=',alen	Length (in words) of the array named by the FITE=,array parameter. The system checks that this length is the length of a file index entry. FITEL=,alen must be specified if FITE=,array is specified.
'SA=',sa	Shared access status. If SA=,sa is omitted, the file is opened only if no other tasks have the file open and no other tasks can open the file until the calling task closes it.
'Y'	Other tasks can open the file for read access.
'N'	Other tasks cannot open the file until the file is closed.

Calling Parameters for USER1 Routines Only

'SADDR=',saddr	Beginning sector address of the file to be opened.
'U1UNIT=',u1	Logical unit number of the device on which the file resides.

Return Parameters

'CONT=',cont	File contiguity.
Y	The file is contiguous.
N	The file is not contiguous.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'EXT=',ext	File extensibility.
Y	The file can be extended.
N	The file cannot be extended.
'FSTO=',fsto	File segment table ordinal (returned only if the file was opened).
'LP=',lp	File duration.
L	Local file.
P	Permanent file.
'MAPS=',maps	Array in which Q5OPEN returns the file's maps.
'MLEN=',mlen	Maximum length (in 512-word blocks) to which the file can be extended. If the file is not opened with write access, the value specified by the LEN= parameter is returned.
'NEWFILE=',nfl	Indicates whether the opened file is a new file.
Y	Q5GETFIL requested the file.
N	Q5GETFIL opened an existing file.

Figure 9-8. Q5GETFIL Call Format (Sheet 3 of 4)

'OCAT=',oc	File ownership.
	PO Pool file.
	PR Private file.
	PU Public file.
'RACS=',racs	Access permission granted to the opened file.
	NO No access.
	R Read access.
	W Write access.
	RW Read and write access.
	T Write-temporary access.
'RDT=',dt	Device type.
	MS Mass storage.
	MT 7-track magnetic tape.
	NT 9-track magnetic tape.
'RFLUN=',rflun	Number SIL assigned to the file.
'RLEN=',len	File length in 512-word blocks.
'RPN=',pn	Six-character identifier of the pack on which the file resides.
'RSLEV=',sl	Security level of the file (0 through 255).
'RTYPE=',typ	File type.
	PD Physical data file.
	VC Virtual code file.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.
'UNIT=',dn	Logical device number for the unit on which the file resides (used by operating system).

Figure 9-8. Q5GETFIL Call Format (Sheet 4 of 4)

Q5GETFIT-GET FIT FIELD VALUES

The user calls the Q5GETFIT routine (refer to figure 9-9) to retrieve the contents of specified fields in a file's FIT. SIL copies the FIT field contents to the variable specified

by the return parameter. The contents are left-justified, blank-filled for character data and right-justified, zero-filled for numerical data (such as buffer lengths).

Call Format

CALL Q5GETFIT({ 'LFN=',lfn } ,optional parameters)

Calling Parameters

'LFN=',lfn File name in the FIT. LFN=',lfn must be specified if FLUN=',rflun is omitted.
'FLUN=',rflun Number SIL assigned to the file and its FIT. FLUN=',rflun must be specified if LFN=',lfn is omitted.

Return Parameters

'ACS=',acs File access permission. SIL can return the following ASCII values.
 W Write access.
 R Read access.
 RW Read and write access.

'BUFL1=',bl1 Buffer one length in 512-word blocks.
'BUFL2=',bl2 Buffer two length in 512-word blocks.
'BUF1=',b1 Array to be used as data buffer one.
'BUF2=',b2 Array to be used as data buffer two.
'BN=',bn Number of the next available block for reading or writing.
'BT=',bt Blocking type. If BT=',bt is omitted, the file has fixed character count blocking.
 C Fixed character count.

'CBP=',ebo Current byte offset within data buffer; used to determine buffer space remaining for get and put I/O calls.

'CFP=',cfp File positioning when the file is closed. If CFP=',cfp is omitted, SIL does not rewind the file.
 N Do not rewind file. Tape files are positioned past the EOF1 label.
 R Rewind file.
 U Unload file.

'CLTYP=',ctp Close type. The type of close operation performed corresponds to the type of open operation performed. SIL can return the following ASCII values.
 R Nonprivileged.
 P Privileged.
 U1 USER1.

'CNF=',cnf Indicates whether file is connected to a terminal. SIL can return the following ASCII values.
 Y The file is connected to a terminal.
 N The file is not connected to a terminal.

Figure 9-9. Q5GETFIT Call Format (Sheet 1 of 4)

'DT=',dt	Device type on which the file resides. SIL can return the following ASCII values. MS Magnetic disk. MT 7-track magnetic tape. NT 9-track magnetic tape.
'ECT=',ect	Number of SIL warning errors issued for the file.
'ERL=',erl	Maximum number of SIL warning errors allowed for the file before SIL aborts the task.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'ES=',es	Last SIL error code for the file.
'FNF=',fnf	Indicates if last SIL error for the file was fatal. SIL can return the following ASCII values. Y The last error was fatal. N The last error was nonfatal.
'LEN=',len	Length in bytes of the data transferred in the last I/O operation on this file.
'LFP=',lfp	Logical file position. SIL can return the following ASCII values. BOI Beginning of information. BOF Beginning of logical file. BOV Beginning of volume. MR Within a logical record. EOR End of logical record. EOG End of group (R and W files only). EOV End of volume. EOF End of file. EOI End of information.
'LLOP=',lop	Last logical operation performed on the file (two-digit number value). The possible values and their meanings are listed in the FIT field description at the beginning of this section.
'MNR=',mnr	Minimum record length in bytes. For record types other than F, SIL checks that the record is not shorter than this value. SIL does not use this parameter when writing F format records.
'MXR=',mxr	Maximum record length in bytes. For F format records, mxr is the fixed record length. For other record types, SIL checks to ensure that the record is not longer than this value.
'OCS=',ocs	Indicates whether file is open or closed. SIL can return the following ASCII values. N The file has never been opened. O The file is open for explicit I/O. I The file is open for implicit I/O. C The file is closed.
'OFP=',ofp	File positioning when the file is opened. SIL can return the following ASCII values. R Rewind the file. N Do not rewind the file.
'PC=',pc	Padding character used to fill the working storage area.
'PEF=',pef	Indicates if a parity error occurred during file I/O. SIL can return the following ASCII values. Y A parity error occurred. N No parity error occurred.
'PTL=',ptl	Partial transfer length in bytes (the amount of data transferred by the last Q5GETP or Q5PUTP call).
'RC=',rc	Number of last full record read or written.

Figure 9-9. Q5GETFIT Call Format (Sheet 2 of 4)

'RFLUN=',rflun	Number SIL assigned to the file.
'RL=',rl	Record length (in bytes) actually transferred.
'RLFN=',lfn	ASCII file name.
'RMF=',rmf	Indicates whether a file has random file organization. Currently, SIL supports only sequential file organization. SIL can return the following ASCII values.
	Y The file is a random file.
	N The file is a sequential file.
'RMK=',rmk	Record delimiting character for R-type records.
'RT=',rt	Record type. SIL can return the following ASCII values.
	F ANSI fixed length.
	R Record mark delimited.
	U Undefined.
	W Control word.
'SFO=',fo	File organization. SIL can return the following ASCII value.
	S Sequential organization.
'SRF=',srf	Indicates whether SIL must complete an I/O request before returning control to the user. SIL can return the following ASCII values.
	Y I/O overlap suppressed.
	N I/O overlap allowed.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.
'UNIT=',dn	Logical device number for the unit on which the file resides (used by operating system).
'WSA=',wsa	Working storage area used by get and put I/O calls.
'WSL=',wsl	Length (in bytes) of the working storage area.
'WPF=',wpf	Indicates whether the last I/O operation was a write operation. SIL can return the following ASCII values.
	Y The last operation was a write.
	N The last operation was not a write.

Return Parameters for Tape Files Only

'DEN=',den	Magnetic tape density. SIL can return the following ASCII values.
	200 200 bpi (7-track tape)
	556 556 bpi (7-track tape)
	800 800 bpi or cpi (7- or 9-track tape)
	1600 1600 cpi (9-track tape)
'EVP=',evp	Volume positioning when SIL senses the end of tape on a volume within a multivolume set. SIL can return the following ASCII values.
	R Rewind the volume.
	N Do not rewind the volume.
	U Unload the volume.
'FID=',fid	17-character tape file identifier from FIT.
'GN=',gn	Four-digit generation number from FIT.
'LT=',lt	Label type. SIL can return the following ASCII values.
	S ANSI standard labeled tape.
	U Unlabeled tape.

Figure 9-9. Q5GETFIT Call Format (Sheet 3 of 4)

'PNO=',pno	Number indicating position of file within multifile set (sequence number).
'RP=',rp	Number of days the file is to be retained after it is written.
'STID=',st	Six-character set identifier.
'TPM=',tpm	Tape mode. SIL can return the following ASCII values.
	BCD Binary coded decimal for 7-track tapes; even parity.
	BIN Unformatted binary for 7- or 9-track tapes; odd parity.
	AS6 6-bit ASCII code for 7-track tape; even parity
	ASC 8-bit ASCII code for 7- or 9-track tape; odd parity.
'VN=',vn	Two-digit version number.
'VSN=',vsn	Six-character volume serial number (VSN) of the tape assigned to this FIT.

Figure 9-9. Q5GETFIT Call Format (Sheet 4 of 4)

Q5GETN-READ PARTITION

The user calls the Q5GETN routine (refer to figure 9-10) to transfer a logical partition (record, group, or file) of data into the working storage area. SIL transfers the data from a physical I/O buffer specified in the FIT. (The Q5OPEN call for the file determines the buffer used.) SIL automatically reads data from mass storage when the buffer is empty.

If the file is positioned within a partition, SIL skips to the beginning of the next partition before transferring data. If the amount of data in the partition exceeds the size of the working storage area, SIL truncates the partition and returns an excess data status code.

Q5GETN transfers partition delimiters of lower-level partitions than the partition specified. For example, if the specified partition is a file, Q5GETN transfers record and group delimiters to the working storage area.

Q5GETN expands compressed R-type records unless the user specifies the NOCOMP parameter on the Q5OPEN call for the file. Blank compression is described under Record Mark Format in section 2.

If the user does not specify a working storage area on the Q5GETN call, SIL uses the working storage area specified in the file's FIT.

SIL recognizes logical partitions as described under Logical Record Formats in section 2.

Q5GETP-READ PARTIAL PARTITION

The user calls the Q5GETP routine (refer to figure 9-11) to transfer part of a logical partition (record, group, or file) of data into the working storage area. SIL transfers the data from physical I/O buffer that it maintains. It automatically reads data into the buffers as needed.

Q5GETP transfers partition delimiters of lower-level partitions than the partition specified. For example, if the specified partition is a file, Q5GETP transfers record and group delimiters to the working storage area.

The user must check for the appropriate end of partition status code to determine when SIL has read the end of partition delimiter. The following are the end of partition codes.

<u>Code</u>	<u>Meaning</u>
1404	End of information.
1440	End of record.
1441	End of group.

If the user specifies the SKIP parameter on the Q5GETP call, SIL skips to the beginning of the next partition before transferring data.

Q5GETP expands compressed R-type records unless the user specifies the NOCOMP parameter on the Q5OPEN call for the file. Blank compression is described under Record Mark Format in section 2.

If the user does not specify a working storage area on the Q5GETP call, SIL uses the working storage area specified in the file's FIT.

Q5GIVE-GIVE FILE OWNERSHIP

The Q5GIVE routine (refer to figure 9-12) transfers ownership of a private file. The file must be closed and must be either a local file or an attached permanent file.

Nonprivileged users can give a file to another user or to a pool. To do so, the user must specify the name or number of the file and either the user number or the pool to which the file is given.

Privileged users can also give a file to the public file list. To do so, the user specifies the name or number of the file and the PUBLIC parameter.

A USER-1 routine can give a file to the input queue manager or to a user. When the input queue manager is given a file, it processes the file as a batch job. To give a file to the input queue manager, the USER-1 routine specifies the name or number of the file, the IQM parameter, and the account under which the job executes. To give a file to a user, a USER-1 routine specifies the name or number of the file, the virtual address and unit of the file, and the user number to which the file is given.

Unless the file is being given to a pool, Q5GIVE cannot transfer ownership of a file that has the same name as an existing public file.

After ownership transfer, the file is an unattached permanent file belonging to the specified owner. The file is no longer attached to the job that issued the Q5GIVE call.

The user can specify a variable rate accounting factor for the file using the VRI parameter. The system provides variable rate accounting for public utilities and application packages that are not to be charged the full rate. The VRI parameter specifies an index into the Variable Rate Table, TVRF. The user should consult a systems analyst for the appropriate index value.

Call Format

CALL Q5GETN({ 'LFN=',lfn
'FLUN=',rflun } ,optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.

'FLUN=',rflun Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.

'PART=',part File partition to be transferred. If PART=,part is omitted, SIL transfers a record.

'R' Record.

'G' Group (W-type and R-type records only).

'F' File.

'WSA=',wsa Working storage area . If WSA=,wsa is omitted, SIL uses the working storage area specified in the file's FIT.

'WSL=',wsl Length of working storage area in bytes. If WSL=,wsl is omitted, SIL uses the working storage length specified in the file's FIT.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'RL=',rl Record length (in bytes) actually transferred.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-10. Q5GETN Call Format

Call Format

CALL Q5GETP({ 'LFN=',lfn }
{ 'FLUN=',rflun } ,optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.

'FLUN=',rflun Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.

'PART=',part File partition. If PART=,part is omitted, SIL transfers a partial record.

 'R' Record.
 'G' Group (W-type and R-type records only).
 'F' File.

'SKIP' Indicates that SIL must transfer data from the beginning of a partition. If the file is positioned within a partition, SIL skips to the next partition delimiter. If 'SKIP' is omitted, SIL transfers data from the current file location.

'WSA=',wsa Working storage area. If WSA=,wsa is omitted, SIL uses the working storage area specified in the file's FIT.

'WSL=',wsl Length of working storage area in bytes. If WSL=,wsl is omitted, SIL uses the working storage length specified in the file's FIT.

Return Parameters

'ERRLEN=',len Error message length is bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'RL=',rl Length (in bytes) of data transferred. This value is the working storage area length unless SIL encountered an error or a partition delimiter.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-11. Q5GETP Call Format

Call Format

CALL Q5GIVE({ 'LFN=',lfn }
{ 'FLUN=',flun }, { 'USER=',un
'POOL=',pool
'PUBLIC'
'IQM','USER=',un, ACCT=,acct } ,optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.

'FLUN=',rflun Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.

'USER=',un User number to which SIL gives the file (if POOL=, PUBLIC, or IQM is not specified). If IQM is specified, USER= specifies the user for which the specified file executes as a batch job.

'VRI=',vri Variable rate index (0 through 255). If VRI=,vri is omitted, SIL sets the variable rate index to zero.

Figure 9-12. Q5GIVE Call Format (Sheet 1 of 2)

Calling Parameters for Pool Files Only

'POOL=',pool Name of the pool to which SIL transfers file ownership. The user must specify either USER=,un, POOL=,pool, or PUBLIC.

'ACS=',acs File access permitted to users. If ACS=,acs is omitted, SIL allows read access.

'N' No access.
'R' Read access.
'W' Write access.
'RW' Read and write access.

Calling Parameters for Privileged Callers

'PUBLIC' Indicates that file ownership is transferred to the public file list. The user must specify either USER=,un, POOL=,pool, or PUBLIC.

'ACS=',acs File access permission. If ACS=,acs is omitted, SIL allows read access.

'N' No access.
'R' Read access.
'W' Write access.
'RW' Read or write access.

'FLAGS=',flg Indicates requested SIL action. If FLAGS=,flg is omitted, SIL performs both the COU and SPB actions.

'COU' Clear the originating user field in the File Index table.
'SPB' Set the file's privileged bit to allow the file, if executed, to issue privileged calls. (The file's controllers do not have privileged status.)
'CS' Perform the actions of both COU and SPB.

'ODIV=',od File's owner division. If ODIV=,od is omitted, the owner is all users.

'ALL' The owner is all users.
'ME' The owner is the originating user.

Calling Parameters for USER-1 Routines

'ACCT=',acct Account number to which the specified batch input file is to belong. ACCT= is specified only if the IQM parameter is specified.

'IQM' SIL gives the specified batch input file to the input queue manager for entry in the input queue. If IQM is specified, the ACCT= and USER= parameters must also be specified and the SADDR= parameter must not be specified.

'SADDR=',adr Beginning sector address of the file, SADDR= is required when a USER-1 routine gives a file to a user. If SADDR= is specified, IQM must not be specified.

'U1UNIT=',ul Logical unit number of the device on which the file resides. U1UNIT= is required when SADDR= is specified.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-12. Q5GIVE Call Format (Sheet 2 of 2)

Q5MAPIN-MAP IN VIRTUAL SPACE

The user calls the Q5MAPIN routine (refer to figure 9-13) to associate virtual region addresses with mass storage addresses. The user specifies as the virtual region an array declared within the program. Q5MAPIN associates the virtual region with the specified mass storage file. The user must have read or write access to the file and must have opened the file for implicit I/O.

SIL, by default, maps into the drop file data areas not mapped in to other files. The user can also specify the drop file on a Q5MAPIN call. Q5MAPIN cannot map in a source file.

After the virtual region is mapped in, references to variables in the virtual region cause the operating system to transfer the data in the mass storage space associated with the referenced variable to central memory. If the user has write access to the file, a Q5MAPOUT call writes the modified data over the original data.

SIL changes the record type in the file's FIT to undefined when it opens the file for implicit I/O. It does not change the record type in the file's file index entry.

Call Format

```
CALL Q5MAPIN('LFN=',lfn, 'FLUN=',rflun, 'VBA=',vba, 'LEN=',ln, optional parameters, 'DROPF')
```

Calling Parameters

'LFN=',lfn	Name of the mass storage file. LFN=,lfn must be specified if FLUN=,rflun and DROPF are omitted.
'FLUN=',rflun	Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn and DROPF are omitted.
'DROPF'	Indicates SIL should map in the drop file. DROPF must be specified if LFN=,lfn and FLUN=,rflun are omitted.
'VBA=',vba	Address of the first word of the array to be mapped in as the virtual region (subscripted array name not entered as a literal). The array must be at least 512 words (one block). VBA=,vba must be specified.
'LEN=',ln	Length in 512-word blocks of the virtual region specified by the VBA=,vba parameter. LEN=,ln must be specified.
'EXT=',ext	File extensibility indicator. If EXT=,ext is omitted, the file is extendible. 'Y' The file is extendible. 'N' The file is not extendible.
'LMA=',lma	Number (relative to the beginning of the file) of the first file sector to be associated with the virtual region. If LMA=,lma is omitted, SIL assumes lma is zero and uses the first sector of the file.
'LPAGE'	Indicates that SIL should map in 128-block units (large pages). If 'LPAGE' is omitted, SIL maps in one-block units (small pages).

Return Parameters

'CONT=',cont	Indicates whether the file is contiguous. SIL can return the following ASCII values. Y The file is contiguous. N The file is segmented.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'EXT=',ext	Indicates whether the file is extendible. SIL returns the following ASCII values. Y The file is extendible. N The file is not extendible.
'MLEN=',mlen	Maximum length of the virtual region (in 512-word blocks). This is the maximum length to which the specified mass storage file can be extended.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-13. Q5MAPIN Call Format

Q5MAPOUT-MAP OUT VIRTUAL SPACE

The user calls the Q5MAPOUT routine (refer to figure 9-14) to disassociate a virtual region from its mass storage space. If the user has write access to the mass storage file, SIL writes the modified data over the original data. If the user does not have write access to the file, SIL discards the modified data at program termination.

The mass storage file mapped out can be one of the user's files, the program's drop file, or the executing program itself (the controllee file).

After a virtual region is mapped out, references to addresses within the region no longer cause the operating system to transfer the corresponding data on mass storage into central memory. However, the program can reference the data that was implicitly read into the region while it was mapped in. Changes to the data are not written on the mass storage file.

Call Format

```
CALL Q5MAPOUT( ( 'LFN=',lfn  
                 'FLUN=',rflun  
                 'DROPF'  
                 'CONTF' ) , 'VBA=',vba, 'LEN=',ln, optional parameters)
```

Calling Parameters

'LFN=',lfn	Name of the mass storage file. LFN=,lfn must be specified if FLUN=,rflun, DROPF, and CONTF are omitted.
'FLUN=',rflun	Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn, DROPF, and CONTF are omitted.
'DROPF'	Indicates SIL should map out the drop file. DROPF must be specified if LFN=,lfn, FLUN=,rflun, and CONTF are omitted.
'CONTF'	Indicates SIL should map out the controllee file. CONTF must be specified if LFN=,lfn, FLUN=,rflun, and DROPF are omitted.
'VBA=',vba	Name of virtual region array to be mapped out. VBA=,vba must be specified.
'LEN=',ln	Length of the virtual region in 512-word blocks. LEN=,ln must be specified.

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-14. Q5MAPOUT Call Format

Q5OPEN-OPEN FILE

The user calls the Q5OPEN routine (refer to figure 9-15) to open a file. Opening a file connects the file to the task for I/O. Q5OPEN can open a file for explicit or implicit I/O or for privileged calls.

If a FIT does not exist for the file, Q5OPEN generates a FIT for the file using parameter specifications and default values. The user can also change the values in the FIT by specifying the appropriate parameters. The changes specified are not permanent because the file's file index entry is not changed.

The user can specify the explicit I/O buffers to be used.

If the file is on magnetic tape, the user must issue a Q5REQUEST call to have the volume mounted on a tape unit. The Q5OPEN call processes the HDR1 label. If SIL encounters an end-of-tape while processing the HDR1 label, it writes the EOVI label, rewinds and unloads the volume, and requests mounting of the next volume, if any, in the multivolume set.

A privileged open call must specify whether other users are to be allowed to concurrently open the file. Other users cannot open the file for write access.

Call Format

Nonprivileged call:

CALL Q5OPEN('LFN=',lfn
'FLUN=',rflun }, optional parameters)

Privileged call:

CALL Q5OPEN('LFN=',lfn
'FLUN=',rflun }, 'SA=',sa, optional parameters)

Calling Parameters

'LFN=',lfn	Name of the file to be opened. LFN=,lfn must be specified if FLUN=,rflun is omitted.
'FLUN=',rflun	Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.
'ACS=',acs	File access desired. If ACS=,acs is omitted, SIL allows read and write access. 'R' Read access. 'W' Write access. 'RW' Read and write access. 'T' Write-temporary access.
'ALEN=',alen	Length in words of the array specified by the MAPS=,map return parameter. This parameter must be specified if the MAPS=,map parameter is specified.
'BT=',bt	Blocking type. If BT=,bt is omitted, the file has fixed character count blocking. 'C' Fixed character count.
'BUFL1=',bl1	Buffer one length in 512-word blocks. If BUFL1=,bl1 is omitted, SIL assumes a buffer length of three blocks.
'BUFL2=',bl2	Buffer two length in 512-word blocks. If BUFL2=,b2 is omitted, SIL assumes a buffer length of three blocks.
'BUF1=',b1	Array to be used as data buffer one. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. This parameter is required if SIL is to read or write the file.
'BUF2=',b2	Array to be used as data buffer two. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. Data buffer two is not required.

Figure 9-15. Q5OPEN Call Format (Sheet 1 of 4)

'EXT=,ext	File extensibility. If EXT=,ext is omitted, the file is extendible. 'Y' The file is extendible. 'N' The file is not extendible.
'IMP'	Indicates that SIL should open the file for implicit I/O. SIL changes the record type in the FIT to undefined; it does not change the record type in the file index entry. If IMP is omitted, SIL opens the file for explicit I/O.
'MNR=,mnr	Minimum record length in bytes. For record types other than F, SIL checks that a record is not shorter than this value. SIL does not use this value when writing F format records. If MNR=,mnr is omitted, SIL assumes the maximum record length is one byte.
'MXR=,mxr	Maximum record length in bytes. For F format records, mxr is the fixed record length. For other record types, SIL checks that the record is not longer than this value. If MXR=,mxr is omitted, SIL assumes the maximum record length is the default set by an installation parameter.
'NOCOMP'	Indicates that SIL should not perform blank compression and expansion on this file. If NOCOMP is omitted, SIL performs blank compression and expansion.
'PC=,pc	Padding character used to fill the working storage area. If PC=,pc is omitted, SIL pads with blanks.
'RMK=,rmk	Record delimiting character for R format records. If RMK=,rmk is omitted, SIL uses the installation-specified character (usually ASCII US, #1F).
'RT=,rt	Record type. If RT=,rt is omitted, SIL assumes the default set by an installation parameter. 'F' ANSI fixed length. 'R' Record mark delimited. 'U' Undefined. 'W' Control word.
'SFO=,fo	File organization. If SFO=,fo is omitted, SIL assumes sequential organization. 'S' Sequential organization.
'WSA=,wsa	Working storage area used by get and put I/O calls.
'WSL=,wsl	Length (in bytes) of the working storage area.

Calling Parameters for Tape Files Only

'ACST=,ast	Accessibility character is recorded in the tape label. If ACST=,ast is omitted, SIL assumes the accessibility character is blank.
'FID=,fid	17-character tape file identifier. If FID=,fid is omitted, SIL uses blanks for the file identifier.
'GN=,gn	Four-digit generation number (1 through 9999). If GN=,gn is omitted, SIL uses blanks for the generation number.
'RP=,rp	Number of days the file is to be retained (0 through 999). SIL determines the expiration date it records in the HDR1 label using the value. If RP=,rp is omitted, SIL retains the file for 30 days.
'STID=,st	Six-character set identifier. The user must specify a set identifier for a file in a multifile set.
'VN=,vn	Two-digit version number (0 through 99). If VN=,vn is omitted, SIL enters '00' as the version number.

Figure 9-15. Q5OPEN Call Format (Sheet 2 of 4)

Calling Parameters for Privileged Users Only

'FITE=',array	Name of array to receive a copy of the file index entry for the opened file. The first word of the array must contain the owner's user number or the name of the pool to which the file belongs. The second word must contain the file name.
'FITLE=',alen	Length (in words) of the array named by the FITE=',array parameter. The system checks that this length is the length of a file index entry.
'FMT'	Indicates that SIL should return a formatted copy of the file index entry. If FMT is omitted, SIL returns an unformatted copy.
'SA=',sa	Shared access status. This parameter is required for privileged open calls. Y Other tasks can open the file for other than write access. N No other task can open the file.

Calling Parameters for USER1 Calls Only

'SADDR=',saddr	Starting sector access of the opened file.
'U1UNIT=',dn	Logical device number of the unit on which the file resides.

Return Parameters

'CONT=',con	Contiguity requirement for the file. SIL returns the following ASCII values. Y The file must be contiguous. N The file can be segmented.
'DT=',dt	Device type on which the file resides. SIL returns the following ASCII values. MS Mass storage. MT 7-track magnetic tape. NT 9-track magnetic tape.
'ERRLEN=',len	Error message length in bytes.
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'EXT=',ext	File extensibility indicator. The system can return the following ASCII values. Y The file is extendible. N The file is not extendible.
'FC=',cat	File category. The system can return the following ASCII values. BA Batch file. MS Mass storage file. OT Output file. SD System-generated drop file. SR Scratch file. UD User-generated drop file. UN Undefined file. WT Write-temporary file.
'FSTO=',fst	File segment table ordinal. If the attempt to open the file failed, SIL does not return an ordinal value.
'LEN=',fl	File length (in 512-word blocks).
'LP=',lp	File duration. The system can return the following ASCII values. L Local (job-duration file). P Permanent file.

Figure 9-15. Q5OPEN Call Format (Sheet 3 of 4)

'MAPS=',map	Array in which SIL returns the file's maps (bound implicit or bound explicit). This parameter must be specified if the ALEN=,alen parameter is specified.
'MLEN=',max	Length (in 512-word blocks) to which SIL can extend the file. When SIL opens a file with read access only, it returns the actual length of the file to max. When SIL opens a file with write access, it returns the maximum length to which it can extend the file.
'OCAT=',own	File ownership category. SIL can return the following ASCII values. PO Pool file. PR Private file. PU Public file.
'PN=',pn	Six-character identifier of the pack on which the file resides.
'RACS=',acs	Access permission granted for the opened file. SIL can return the following ASCII values. N No read or write access. R Read access only. W Write access only. RW Read and write access. T Write-temporary access.
'REXT=',rext	File extendibility. SIL returns the following ASCII values. Y The file is extendible. N The file is not extendible.
'SLEV=',slev	Security level of the file (0 through 255).
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.
'TYPE=',typ	File type. The system can return the following ASCII values. PD Physical data file. VC Virtual code file.
'UNIT=',dn	Logical device number of the unit on which the file resides.

Figure 9-15. Q5OPEN Call Format (Sheet 4 of 4)

Q5PATACH-ATTACH POOL

The user calls the Q5PATACH routine (refer to figure 9-16) to access the files in an existing pool. To access a pool, the user must be either the pool boss or a user granted access by the pool boss. The type of file access allowed is that specified when the file was defined.

The task can have up to four pools of files attached concurrently.

Q5PCREAT-CREATE POOL

The user calls the Q5PCREAT routine (refer to figure 9-17) to create a pool. Q5PCREAT adds the specified name to the pool list. Initially, no files belong to the pool. After creating the pool, the user can issue a Q5GIVE call to give files to the pool. The user who creates the pool becomes the pool boss.

Call Format

CALL Q5PATACH('POOL=',pool, optional parameters)

Calling Parameters

'POOL=',pool Name of pool to be attached. The name must be left-justified and blank-filled. This parameter is required.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-16. Q5PATACH Call Format

Call Format

CALL Q5PCREAT('POOL=',pool, optional parameters)

Calling Parameters

'POOL=',pool Name of the new pool. The name must be eight letters and digits, starting with a letter, left-justified and blank-filled. This parameter is required.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-17. Q5PCREAT Call Format

Q5PDESTR-DESTROY POOL

The user calls the Q5PDESTR routine (refer to figure 9-18) to remove the specified pool name from the pool list. The user must be the pool boss, no files can belong to the pool, and the pool cannot be attached to a task (including the task issuing the Q5PDESTR call).

Q5PDTACH-DETACH POOL

The user calls the Q5PDTACH routine (refer to figure 9-19) to return an attached pool of files. All files in the pool need not be closed before the user returns the pool.

Q5PGRACC-GRANT ACCESS TO POOL

The user calls the Q5PGRACC routine (refer to figure 9-20) to grant other users access to a pool. The user must be the pool boss for the specified pool.

Q5PGRACC can grant access to all users or to a specified list of users. The type of file access allowed is that specified when the file was defined.

Q5POOLS-LIST POOLS

The entry format for the Q5POOLS routine is defined in figure 9-21. The user calls the Q5POOLS routine (refer to figure 9-22) to obtain a list of all pools and pool bosses. Q5POOLS copies the list into the specified buffer. It copies a two-word entry for each pool.

Q5POOLS can return the number of words copied to the buffer. The user must divide this number by two to obtain the number of pool entries copied.

Call Format

CALL Q5PDESTR('POOL=',pool, optional parameters)

Calling Parameters

'POOL=',pool Name of the new pool. The name must be left-justified and blank-filled. This parameter is required.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-18. Q5PDESTR Call Format

Call Format

CALL Q5PDTACH('POOL=',pool, optional parameters)

Calling Parameters

'POOL=',pool Pool name, left-justified and blank-filled. This parameter is required.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-19. Q5PDTACH Call Format

Call Format

CALL Q5PGRACC('POOL=',pool, optional parameters)

Calling Parameters

'POOL=',pool Pool name, left-justified and blank-filled. This parameter is required.

'NU=',nu Number of user numbers in the list specified by the ULIST=,ul parameter. If NU=,nu is omitted, SIL assumes no user numbers are specified by ULIST=,ul.

'ULIST=',ul List of user numbers to be granted access to the pool. The numbers must be in integer format, one user number per word. If ULIST=,ul is omitted, SIL grants all user numbers access to the pool.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-20. Q5PGRACC Call Format

pcount	16	res	16	pfree	12	pboss	20
poolname							64

pcount	Number of users that have the pool attached.
res	Reserved.
pfree	Currently unused.
pboss	User number of pool boss (binary format).
poolname	Pool name (eight ASCII characters).

Figure 9-21. Pool List Entry Format

Call Format

CALL Q5POOLS('BUFFER=',bfr,'BUFLLEN=',bl,optional parameters)

Calling Parameters

'BUFFER=',bfr Array to which the pool and pool boss names are copied. This parameter is required.

'BUFLLEN=',bl Length of the array specified by the BUFFER=,bfr parameter. This parameter is required.

Return Parameters

'BUFLLEN=',bl Number of words copied to the buffer. This parameter is also a required calling parameter.

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-22. Q5POOLS Call Format

Q5PREACC-REMOVE ACCESS TO POOL

The user calls the Q5PREACC routine (refer to figure 9-23) to remove pool access granted to one or more users. The user calling Q5PREACC must be the pool boss for the specified pool.

The pool cannot be attached to a user number when SIL processes a Q5PREACC call for the pool.

Q5PURGE-PURGE FILE

The user calls the Q5PURGE routine (refer to figure 9-24) to purge a permanent file. Q5PURGE deletes the file index

entry for the specified file and releases its mass storage space. SIL assumes the file is the private file with the specified name unless the OCAT parameter on the Q5PURGE call specifies that the file is a pool or public file. To purge a pool file, the user must be privileged or be the pool boss and must have attached the pool to the task. To purge a public file, the user must be privileged.

A privileged user can purge other users' private and pool files.

The file must be closed before it is purged. If the purged file is attached to a job, it becomes a local file.

Call Format

CALL Q5PREACC('POOL=',pool,'ULIST=',ul,optional parameters)

Calling Parameters

'POOL=',pool Name of the pool from which SIL should remove access privileges. The pool name must be left-justified and blank-filled in the name. This parameter is required.

'ULIST=',ul List of user numbers whose access privileges SIL should remove. The user numbers must be in binary (right-justified, zero-filled). This parameter is required.

'NU=',nu Number of user numbers specified by ULIST=',ul. If NU=',nu is omitted, SIL assumes ULIST=',ul specifies one user number.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-23. Q5PREACC Call Format

Call Format

Nonprivileged call:

CALL Q5PURGE({ 'LFN=',lfn }
{ 'FLUN=',rflun }, optional parameters)

Privileged Call:

CALL Q5PURGE({ 'LFN=',lfn }
{ 'FLUN=',rflun }, 'OWNER=',un, optional parameters)

Calling Parameters

'LFN=',lfn Name of a permanent file. LFN=,lfn must be specified if FLUN=,rflun is omitted.

'FLUN=',rflun Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.

'OCAT=',oc File ownership category. If OCAT=,oc is omitted, SIL purges a private file.

 'PO' Pool file.

 'PR' Private file.

 'PV' Public file.

Calling Parameter for Privileged Users Only

'OWNER=',own User number or pool name to which the file belongs. A user number must be an integer, right-justified and zero-filled; a pool name must be a character string, left-justified with blank fill.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-24. Q5PURGE Call Format

Q5PUSERL-LIST USERS WITH ACCESS TO POOL

The user calls the Q5PUSERL routine (refer to figure 9-25) to obtain a list of users having access to a pool. The user must be the pool boss of the specified pool.

Q5PUSERL copies the user numbers to the specified buffer, one user number per word in integer format.

Q5PUTN-WRITE PARTITION

The user calls the Q5PUTN routine (refer to figure 9-26) to transfer data from the working storage area to a physical I/O buffer. Q5PUTN appends a partition delimiter to the data. The Q5OPEN call for the file determines the buffer used. SIL automatically writes the data to mass storage when the buffer is full.

If the user does not specify a working storage area on the Q5PUTN call, SIL uses the working storage area specified in the file's FIT. Q5PUTN transfers the amount of data specified on the working storage area length.

For F format records, Q5PUTN transfers the number of bytes specified in the mxr (maximum record length) field of the FIT. For other record formats, Q5PUTN transfers the number of bytes of data specified in the wsl (working storage length) field, although SIL checks that wsl is within the minimum and maximum lengths specified by the mnr and mxr fields.

If the file is an R-format file, Q5PUTN compresses consecutive blanks within the data unless the user specifies the NOCOMP parameter on the Q5OPEN call for the file. Blank compression is described under Record Mark Format in section 2.

Call Format

CALL Q5PUSERL('POOL='pool,'ULIST='ul,'NU='nu,optional parameters)

Calling Parameters

'POOL='pool Name of pool whose access list SIL is to supply. The name must be left-justified and blank-filled. This parameter is required.

'NU='nu Number of words in the buffer specified by the ULIST='ul parameter. This parameter is required.

Return Parameters

'NU='nu Number of user numbers copied to the buffer specified by 'ULIST='ul.

'ERRLEN='len Error message length in bytes (integer).

'ERRMSG='msg Error message. The variable msg must be 80 bytes long.

'STATUS='stat Status code. Refer to SIL Error Processing in section 8 for more information.

'ULIST='ul Buffer to contain the list of user numbers having access to the specified pool, one user number per word. This parameter is required.

Figure 9-25. Q5PUSERL Call Format

Call Format

CALL Q5PUTN({ 'LFN=',lfn }
{ 'FLUN=',rflun }, optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.

'FLUN=',rflun Number SIL assigned to the file and its FIT. FLUN=,rflun must be specified if LFN=,lfn is omitted.

'PART=',part File partition delimiter. If PART=,part is omitted, SIL appends a record delimiter to the data.

 'R' Record.
 'G' Group.
 'F' File.

'WSA=',wsa Working storage area. If WSA=,wsa is omitted, SIL uses the working storage area specified in the file's FIT.

'WSL=',wsl Number of bytes of data transferred (working storage area length). If WSL=,wsl is omitted, SIL uses the working storage length specified in the file's FIT.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-26. Q5PUTN Call Format

Q5PUTP-WRITE PARTIAL PARTITION

The user calls the Q5PUTP routine (refer to figure 9-27) to transfer data from the working storage area to a physical I/O buffer. The Q5OPEN call for the file determines the buffer used. SIL automatically writes the data to mass storage when the buffer is full.

The user calls Q5PUTP to transfer partial records or undefined records (U format). Q5PUTP does not add a partition delimiter to the data unless the user specifies the TERM parameter. The user can also add a partition delimiter by issuing a Q5ENDPAR call.

If the user does not specify a working storage area on the Q5PUTP call, SIL uses the working storage area specified in the FIT. Q5PUTP transfers the amount of data specified as the working storage length.

If the file is an R-format file, Q5PUTP compresses consecutive blanks within the data unless the user specifies the NOCOMP parameter on the Q5OPEN call for the file. Blank compression is described under Record Mark Format in section 2.

Call Format

CALL Q5PUTP({ 'LFN=',lfn
'FLUN=',rflun }, optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.

'FLUN=',rflun Number SIL assigned to the file and its FIT. FLUN=,rflun must be specified if LFN=,lfn is omitted.

'PART=',part File partition delimiter appended if the user specifies the TERM parameter. If PART=,part is omitted, SIL appends a record delimiter.

 'R' Record.
 'G' Group.
 'F' File.

'TERM' Indicates that SIL should append a partition delimiter to the data transferred. If TERM is omitted, a partition delimiter is not appended.

'WSA=',wsa Working storage area. If WSA=,wsa is omitted, SIL uses the working storage area specified in the file's FIT.

'WSL=',wsl Number of bytes of data transferred (working storage area length). If WSL=,wsl is omitted, SIL uses the working storage length specified in the file's FIT.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-27. Q5PUTP Call Format

Q5READ-READ BLOCK

The user calls the Q5READ routine (refer to figure 9-28) to transfer one or more blocks of data from a file to a program buffer. The user must attach the file and open it for explicit I/O before issuing the Q5READ call. The user can specify the buffer to be used on the Q5READ call; otherwise, SIL uses buffer one as specified in the FIT.

Unless the user specifies the WAIT parameter, SIL returns control to the caller immediately after it issues the request (before the data transfer is complete). The user should check for completion of the data transfer with a Q5CHECK call; however, if the user specifies the WAIT parameter, SIL does not return control to the caller until after the data transfer is complete.

If the user specifies a buffer on a Q5READ call using the BUFFER= parameter, the buffers specified in the FIT are no longer defined. Subsequent Q5READ and Q5WRITE calls must specify the BUFFER= parameter.

SIL does not check whether program I/O buffers overlap. For instance, one buffer could extend from address 1 to address 512 and another buffer extend from address 100 to address 612. In this case, a read to the second buffer after a read to the first buffer would overwrite the last 412 words of data read into the first buffer.

For a magnetic tape file, only one I/O request can be pending at a time; therefore, if the file is on tape, the user must check that the Q5READ data transfer is complete before issuing another Q5READ call specifying the file.

If, while reading data from a tape file, SIL reads the EOVI label or end of tape indicator, it performs one of the following actions depending on the contents of the vsn and STID fields in the FIT.

<u>vsn</u>	<u>stid</u>	<u>SIL Action</u>
Blank	Blank	Returns an end of tape status code.
Blank	Nonblank	Requests the operator assign an available tape volume to the file.
Nonblank	Blank	Returns an end of tape status code.
Nonblank	Nonblank	Requests the operator mount the next volume in the VSN list; if all volumes on the list have been read, SIL returns an end of tape status code.

After SIL returns an end of tape status code, the user must rewind or reposition the tape before issuing another Q5READ call for the file.

Call Format

CALL Q5READ({ 'LFN=',lfn
'FLUN=',rflun }, optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.

FLUN=',rflun Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.

'BUFLEN=',bfl Length of data buffer in 512-word blocks.

'BUF1' Indicates SIL should use buffer number one as specified in the FIT. If BUF1 is omitted and either BUFFER=,bfr or BUF2 is specified, SIL uses the specified buffer. If BUFFER=,bfr, BUF1, and BUF2 are omitted, SIL uses buffer number one as specified in the FIT.

'BUF2' Indicates SIL should use buffer number two as specified in the FIT. If BUF2 is omitted and BUFFER=,bfr is specified, SIL uses the specified buffer. If BUF2 and BUFFER=,bfr are omitted, SIL uses buffer number one as specified in the FIT.

'WAIT' Indicates SIL should wait for completion of this read request before returning control to the caller. If WAIT is omitted, SIL returns control immediately to the caller.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'LEN=',rl Number of bytes transferred. If the operation is not complete, the value is undefined.

'RSN=',rsn Number assigned to the request. A Q5CHECK call uses this identifier.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Return Parameters for Tapes Only

'XBC=',xbc Excess bits read. If SIL returns a nonzero value after reading a tape written by a CYBER 200 system, the byte count that Q5READ returns via the LEN=,rl parameter must be decremented by one.

Figure 9-28. Q5READ Call Format

Q5REDUCE-REDUCE FILE SPACE

The user calls the Q5REDUCE routine (refer to figure 9-29) to reduce the length of a file to the length of the data in the file. The Q5REDUCE releases mass storage space allocated to the file which has addresses higher than the highest address accessed in the file.

SIL allocates mass storage space for a file as one or two segments. After that space is filled, it can extend the file two or three times (up to four segments). A Q5REDUCE call does not increase the number of times SIL can extend the file or increase the space it can add to the file.

Q5RETFIT-RETURN FIT

The user calls the Q5RETFIT routine (refer to figure 9-30) to return a FIT. The file associated with the FIT must be closed before its FIT is returned. The user must generate a new FIT for the file before SIL can again read or write the file.

The RETFIT parameter on the Q5CLOSE or Q5RETURN call can also return the FIT. SIL returns all FITs when the task completes.

No more than 110 FITs can be concurrently associated with a task.

Call Format

CALL Q5REDUCE ({ 'LFN=',lfn
'FLUN=',rflun } , optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=,lfn must be specified if FLUN=,rflun is omitted.
'FLUN=',rflun Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.
'NEWLEN=',nfl New file length in 512-word blocks. If NEWLEN=,nfl is omitted, the file extends to the word with the highest address accessed.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).
'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.
'LEN=',len The new length of the file in 512-word blocks.
'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-29. Q5REDUCE Call Format

Call Format

CALL Q5RETFIT ({ 'LFN=',lfn
'FLUN=',rflun } , optional parameters)

Calling Parameters

'LFN=',lfn File name in FIT. LFN=,lfn must be specified if FLUN=,rflun is omitted.
'FLUN=',rflun Number SIL assigned to the FIT. FLUN=,rflun must be specified if LFN=,lfn is omitted.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).
'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.
'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-30. Q5RETFIT Call Format

Q5RETURN-RETURN FILE

The user calls the Q5RETURN routine (refer to figure 9-31) to return a file. A returned local file no longer exists; SIL releases its mass storage space. A returned permanent file is detached from the job. A file must be closed before it is returned.

Returning a file does not return the file's FIT unless the user specifies the RETFIT parameter.

When the file specified on the Q5RETURN call is a tape file, SIL rewinds and unloads the volume. After this operation, the tape unit is available for assignment to another tape file.

Q5REWIND-REWIND FILE

The user calls the Q5REWIND routine (refer to figure 9-32) to position a file at its beginning of information. If the last operation SIL performed on the file was a write operation, Q5REWIND writes an end-of-information indicator before rewinding the file.

For tape files, Q5REWIND rewinds the current volume mounted. If the last I/O operation SIL performed on the file was a write operation, Q5REWIND writes an EOF1 label before rewinding the tape. Q5REWIND positions the tape at the beginning of the file if it is on the current volume; if the beginning of the file is on another volume, Q5REWIND positions the file at the beginning of the current volume.

Call Format

CALL Q5RETURN(('LFN=',lfn
'FLUN=',rflun), optional parameters)

Calling Parameters

'LFN=',lfn	Name of file to be returned. LFN=,lfn must be specified if FLUN=,rflun is omitted.				
'FLUN=',rflun	Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.				
'RETFIT'	Indicator that SIL should discard the FIT for the file. If RETFIT is omitted, SIL does not discard the FIT.				
'UNLOAD=',x	Indicates whether SIL should unload a tape file. <table><tbody><tr><td>'N'</td><td>Do not unload file.</td></tr><tr><td>'Y'</td><td>Unload file.</td></tr></tbody></table>	'N'	Do not unload file.	'Y'	Unload file.
'N'	Do not unload file.				
'Y'	Unload file.				

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-31. Q5RETURN Call Format

Call Format

CALL Q5REWIND(('LFN=',lfn
'FLUN=',flun), optional parameters)

Calling Parameters

'LFN=',lfn	Name of file to be rewound. LFN=,lfn must be specified if FLUN=,rflun is omitted.
'FLUN=',rflun	Number SIL assigns to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-32. Q5REWIND Call Format

Q5ROUTE-ROUTE FILE

The user calls the Q5ROUTE routine (refer to figure 9-33 to specify a file destination. The file must be a local file or an attached permanent file. It becomes an unattached permanent file at its destination. Its FIT is not destroyed.

File routing can be immediate or deferred. (Routing to the input queue cannot be deferred.) If the user specifies the

DEFER parameter on the Q5ROUTE call, SIL stores the routing specifications in the file index entry, but does not route the file until the file is released. The file is released by completion of the task or by a Q5GIVE call or another Q5ROUTE call specifying the file. A Q5ROUTE call releasing a deferred file can specify new routing specifications for the file.

Call Format

CALL Q5ROUTE('LFN=',lfn
'FLUN=',rflun),optional parameters)

Calling Parameters

'LFN=',lfn File name. LFN=',lfn must be specified if FLUN=',rflun is omitted.

'FLUN=',rflun Number SIL assigned to the file. FLUN=',rflun must be specified if LFN=',lfn is omitted.

'CM=',cm Conversion mode. If CM=',cm is omitted, SIL uses the system default value.

 'BF' Binary.
 'DI' Display code (64-character set).
 'EC' Extended display code (128-character set).

'DC=',dc Disposition code. If DC=',dc is omitted, SIL uses the system default value.

 'IN' Input to batch processor.
 'LR' Print on 580-12 printer on the front-end processor.
 'LS' Print on 580-16 printer on the front-end processor.
 'LT' Print on 580-20 printer on the front-end processor.
 'PF' Store as permanent file.
 'PR' Print on any available line printer.
 'PU' Punch file.
 'P1' Print on 501 printer on the front-end processor.
 'P2' Print on 512 printer on the front-end processor.
 'SC' Discard file at end of task.

'DEFER' Indicator that file disposition is to be deferred. If DEFER is omitted, SIL performs the file disposition immediately.

'DI=',di Eight characters (from the display code 64-character set) to be printed on the banner page at the front-end processor. If DI=',di is omitted, SIL uses the system default value.

'EC=',ec Print or punch format. If EC=',ec is omitted, SIL uses the system default value.

 '26' O26 keypunch.
 '29' O29 keypunch.
 '80' 80-column binary.

 Files printed at the front-end processor use the following values.

 'A4' ASCII 48-character
 'A6' ASCII 64-character
 'A9' ASCII 95-character
 'B4' BCD 48-character
 'B6' BCD 64-character

'IC=',ic File format. If IC=',ic is omitted, SIL uses the system default value.

 'AS' 8-bit ASCII code; ANSI carriage control if print file.
 'BI' Binary.
 'PA' 8-bit ASCII code; ASCII carriage control if print file.

'NAC=',nac Access station area code. If NAC=',nac is omitted, SIL uses the system default value.

Figure 9-33. QROUTE Call Format (Sheet 1 of 2)

'OQNAME=',oq	Five characters to identify the file in the output queue. The first character must be a letter. The system adds two unique job sequence characters as the sixth and seventh characters. The eighth character is a blank. If OQNAME=,oq is omitted, SIL uses the system default value.
'OT=',ot	Origin type for a file destined for the access station. If OT=,ot is omitted, SIL uses the system default value.
'ST=',sid	Site identifier identifying the processor to which SIL routes the file for execution ('DC=','IN') or output. If ST=,sid is omitted, SIL uses the system default value. Except for the following identifiers, the installation determines the site identifiers. 'AST' Access station. 'URS' Unit record station.
'TID=',tid	Terminal identifier. For files destined for the access station, tid is a one to seven-character user number of a logged-in user. For files destined for the central site (not a terminal), tid is zero. For files destined for the CYBER 200 link station, tid is a two-character alphanumeric terminal identifier. If TID=,tid is omitted, SIL uses the system default value.
Return Parameters	
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-33. QROUTE Call Format (Sheet 2 of 2)

Q5RQUEST-REQUEST LOCAL OR TAPE FILE

The user calls the Q5RQUEST routine (refer to figure 9-34) to create a local file or to create or access a tape file. SIL allocates mass storage space at job termination. The created file is closed.

The Q5RQUEST call specifying a file must be the first reference to the file within a task. A FIT cannot exist for the file before the Q5RQUEST call.

To create or access a tape file, the user must specify the DT parameter on the Q5RQUEST call. The system assigns a tape unit to the file and the operator mounts the specified volume. If the user omits the VSN parameter, the operator mounts an available tape. If the user specified the NEVLAB parameter, Q5RQUEST writes a new VOL1 label.

If the user specified the NOLABEL parameter, Q5RQUEST positions the tape at load point. If the user omits the NEVLAB and NOLABEL parameters Q5RQUEST checks the VOL1 label to determine if the operator mounted the correct volume.

A Q5RQUEST call sets the following FIT fields to their default values.

CFP	N (no rewind).
EVP	U (unload).
FID	Blanks (file id).
GN	Blanks (generation).
RP	30 (retention period in days).
VN	0 (version).

To change the values of these FIT fields, the user must issue a Q5SETFIT call after the Q5RQUEST call.

Call Format

CALL Q5RQUEST('LFN=',lfn, optional parameters)

Calling Parameters

'LFN=',lfn	File name. This parameter is required.
'ACS=',acs	File access permission. If ACS=, acs is omitted, SIL allows read and write access. 'W' Write access. 'R' Read access. 'RW' Read and write access.
'BT=',bt	Blocking type. If BT=,bt is omitted, the file has fixed character count blocking. 'C' Fixed character count.
'DT=',dt	Device type on which the file resides. If DT=,dt is omitted, the file resides on mass storage. 'MS' Mass storage. 'MT' 7-track magnetic tape. 'NT' 9-track magnetic tape.
'EXT=',ext	File extensibility indicator. If EXT=,ext is omitted, the file is extendable. 'Y' The file is extendable. 'N' The file is not extendable.
'FC=',fc	File category. If FC=,fc is omitted, the file is a user file. 'B' Batch file (batch processor controller). 'U' User file.
'LEN=',fl	File length is 512-word blocks. If LEN=,fl is omitted, the file is eight 512-word blocks.
'MNR=',mnr	Minimum record length in bytes. For record types other than F, SIL checks that the record is not shorter than this value. SIL does not use this parameter when writing F-type records. If MNR=,mnr is omitted, SIL assumes the minimum record length is one byte.
'MXR=',mxr	Maximum record length in bytes. For F format records, mxr is the fixed record length. For other record formats, SIL checks that the record is not longer than this value. If MXR=,mxr is omitted, SIL assumes the maximum record length is the default set by an installation parameter.
'NOSEG'	Indicates that file must be contiguous (not written in segments). If NOSEG is omitted, SIL can segment the file.
'PC=',pc	Padding character used to fill the working storage area. If PC=,pc is omitted, SIL pads with blanks.
'PN=',pn	Six-character identifier of the disk pack on which SIL creates the file. If PN=,pn is omitted, the system assigns mass storage space for the file.
'RMK=',rmk	Record delimiting character for R-type records. If RMK=,rmk is omitted, SIL uses the installation-specified character (usually ASCII US, #1F code).
'RT=',rt	Record type. If RT=,rt is omitted, SIL assumes the default set by an installation parameter. 'F' ANSI fixed length. 'R' Record mark delimited. 'U' Undefined. 'W' Control word.
'SFO=',fo	File organization. If SFO=,fo is omitted, SIL assumes sequential organization. 'S' Sequential organization.
'SLEV=',sl	Security level, (1 through 255, but less than or equal to that of the caller). If SLEV=,sl is omitted, SIL sets the file security level equal to that of the caller.

Figure 9-34. Q5RQUEST Call Format (Sheet 1 of 2)

'TYPE=',typ	File type. If TYPE=,typ is omitted, SIL assumes the file is a physical data file.
'PD'	Physical data file.
'VC'	Virtual code file.
<u>Calling Parameters for Tapes Only</u>	
'DEN=',den	Tape density. If DEN=,den is omitted, SIL assumes 1600 cpi.
'200'	200 bpi (7-track tape).
'556'	556 bpi (7-track tape).
'800'	800 bpi or cpi (7- or 9-track tape).
'1600'	1600 cpi (9-track tape).
'NEWLAB'	Indicator that SIL should write new labels on the tape. If NEWLAB is omitted, SIL reads the existing label.
'NOLABEL'	Indicates that the tape is unlabeled. If NOLABEL is omitted, SIL assumes ANSI standard labels.
'OWNER=',own	14-character owner identification, left-justified and blank-filled. If OWNER=,own is omitted, SIL uses blanks as the owner identification.
'STID=',st	Six-character set identifier. The user must specify a set identifier for a file in a multifile set.
'TPM=',tpm	Tape mode. If TPM=,tpm is omitted, SIL reads or writes tape data in 8-bit ASCII character code.
'BCD'	Binary coded decimal for 7-track tapes; even parity.
'BIN'	Unformatted binary for 7- or 9-track tapes; odd parity.
'AS6'	6-bit ASCII code for 7-track tape; even parity.
'ASC'	8-bit ASCII code for 7- or 9-track tape; odd parity.
'TRY=',try	Error recovery and noise handling. If TRY=,try is omitted, SIL attempts error recovery, but does not process noise records.
'SR'	Attempts error recovery, but does not process noise records.
'NR'	Does not attempt error recovery or process noise records.
'SRN'	Attempts error recovery and processes noise records.
'NRN'	Does not attempt error recovery, but processes noise records.
'VSN=',vsn	Six-character volume serial number (VSN) of the requested tape. If VSN=,vsn is omitted or the specified VSN is zero, the operator assigns a tape to the file.
'VSNL=',vsnl	Number of volume serial numbers in the vsn array. If VSN=,vsn is omitted, SIL assumes the array contains one VSN.
<u>Return Parameters</u>	
'CONT=',con	Initial contiguity of the mass storage file.
Y	The file space is contiguous.
N	The file space is segmented.
'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg must be 80 bytes long.
'MLEN=',max	Maximum length of the file in 512-word blocks (refer to File Space Allocation in section 2).
'FLUN=',rflun	Number SIL assigned to the open file.
'RPN=',pn	Six-character identifier of the disk pack on which the file resides.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.
'UNIT=',dn	Logical device number for the unit on which the file resides.

Figure 9-34. Q5RQUEST Call Format (Sheet 2 of 2)

Q5SETFIT-SET FIT FIELD VALUES

The user calls the Q5SETFIT routine (refer to figure 9-35) to change the values of specified FIT fields. The FIT must already exist. The values in the fields not specified are not changed.

Q5SKIP-SKIP PARTITION

The user calls the Q5SKIP routine (refer to figure 9-36) to position a file. Q5SKIP can set the current file position forward or backward a specified number of records, groups, files, or blocks.

If the user requests a backward skip and the last I/O operation SIL performed on the file was a write operation, Q5SKIP writes end-of-file and end-of-information indicators before positioning the file. The user cannot request a forward skip if the last I/O operation was a write operation.

The only file positioning possible for a U format file is skipping by blocks.

Skipping of files on magnetic tape is equivalent to skipping by tape marks; therefore, skipping files is not recommended for standard labeled tapes, as tape labels are delimited by tape marks.

Call Format

CALL Q5SETFIT ('LFN=',lfn , 'FLUN=',rflun , optional parameters)

Calling Parameters

'LFN=',lfn	Name of a permanent file. LFN=,lfn must be specified if FLUN=,rflun is omitted.
FLUN=,rflun	Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.
'ACS=',acs	File access permission. If ACS=,acs is omitted, SIL allows read and write access. 'W' Write access. 'R' Read access. 'RW' Read and write access.
'BN=',bn	Number of the next available block for reading or writing.
'BT=',bt	Blocking type. If BT=,bt is omitted, the file has fixed character count blocking. 'C' Fixed character count.
'BUFL1=',bl1	Buffer one length in 512-word blocks. If BUFL1=,bl1 is omitted, SIL assumes a buffer length of three blocks.
'BUFL2=',bl2	Buffer two length in 512-word blocks. If BUFL2=,bl2 is omitted, SIL assumes a buffer length of three blocks.
'BUF1=',b1	Array to be used as data buffer one. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. This parameter is required if SIL is to read or write the file.
'BUF2=',b2	Array to be used as data buffer two. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. Data buffer two is not required.
'CFP=',cfp	File positioning when the file is closed. If CFP=,cfp is omitted, SIL does not rewind the file. 'N' Do not rewind file. Tape files are positioned past the EOF1 label. 'R' Rewind file. 'U' Unload file.
'ERL=',erl	Maximum number of SIL warning errors allowed for the file before SIL aborts the task. If a zero limit is specified or ERL=,erl is omitted, SIL allows an unlimited number of warning errors.
'MNR=',mnr	Minimum record length in bytes. For record types other than F, SIL checks that a record is not shorter than this value. SIL does not use this value when writing F format records. If MNR=,mnr is omitted, SIL assumes the minimum record length is one byte.

Figure 9-35. Q5SETFIT Call Format (Sheet 1 of 3)

'MXR=',mxr	Maximum record length in bytes. For F format records, mxr is the fixed record length. For other record types, SIL checks that the record is not longer than this value. If MXR=,mxr is omitted, SIL assumes the maximum record length is the default set by an installation parameter.
'NOCOMP=',nc	Indicates whether SIL should perform blank compression and expansion on this file. If NOCOMP is omitted, SIL compresses and expands consecutive blanks for an R format file as described in section 2. 'Y' Perform blank compression. 'N' Do not perform blank compression.
'OFP=',ofp	File positioning when the file is opened. If OFP=,ofp is omitted, SIL rewinds the file. 'R' Rewind the file. 'N' Do not rewind the file.
'PC=',pc	Padding character used to fill the working storage area. If PC=,pc is omitted, SIL pads with blanks.
'RMK=',rmk	Record delimiting character for R format records. If RMK=,rmk is omitted, SIL uses the installation-specified character (usually ASCII US, #1F code).
'RT=',rt	Record type. If RT=,rt is omitted, SIL assumes the default set by an installation parameter. 'F' ANSI fixed length. 'R' Record mark delimited. 'U' Undefined. 'W' Control word.
'SFO=',fo	File organization. If SFO=,fo is omitted, SIL assumes sequential organization. 'S' Sequential organization.
'SRF=',srf	Indicates that SIL must complete an I/O request before returning control to the caller. If SRF=,srf is omitted, SIL can return control to the caller before completing the read or write. 'Y' Suppress overlapped I/O. 'N' Allow overlapped I/O.
'WSA=',wsa	Working storage area used by get and put calls.
'WSL=',wsl	Length (in bytes) of the working storage area.

Calling Parameters for Tape Files Only

'DEN=',den	Tape density. If DEN=,den is omitted, SIL assumes 1600 cpi. '200' 200 bpi (7-track tape). '556' 556 bpi (7-track tape). '800' 800 bpi or cpi (7- or 9-track tape). '1600' 1600 cpi (9-track tape).
'EVP=',evp	Volume positioning when SIL senses the end of tape on a volume within a multivolume set. If EVP=,evp is omitted, SIL unloads the volume. 'R' Rewind the volume. 'N' Do not rewind the volume. 'U' Unload the volume.
'FID=',fid	17-character tape file identifier. If FID=,fid is omitted, SIL uses blanks for the file identifier field.
'GN=',gn	Four-digit generation number (1 through 9999). If GN=,gn is omitted, SIL uses blanks for the generation number.
'LT=',lt	Label type. If LT=,lt is omitted, SIL assumes ANSI standard labels. 'S' ANSI standard labeled tape. 'U' Unlabeled tape.

Figure 9-35. Q5SETFIT Call Format (Sheet 2 of 3)

'RP=',rp	Number of days the file is to be retained (0 through 999). SIL determines the expiration date it records in the HDR1 label using this value. If RP=,rp is omitted, SIL retains the file for 30 days.								
'STID=',st	Six-character set identifier. The user must specify a set identifier for a file in a multifile set.								
'TPM=',tpm	Tape mode. If TPM=,tpm is omitted, SIL reads or writes tape data in 8-bit ASCII character code. <table border="0"> <tr> <td>'BCD'</td><td>Binary coded decimal for 7-track tapes; even parity.</td></tr> <tr> <td>'BIN'</td><td>Unformatted binary for 7- or 9-track tapes; odd parity.</td></tr> <tr> <td>'AS6'</td><td>6-bit ASCII code for 7-track tape; even parity.</td></tr> <tr> <td>'ASC'</td><td>8-bit ASCII code for 7- or 9-track tape; odd parity.</td></tr> </table>	'BCD'	Binary coded decimal for 7-track tapes; even parity.	'BIN'	Unformatted binary for 7- or 9-track tapes; odd parity.	'AS6'	6-bit ASCII code for 7-track tape; even parity.	'ASC'	8-bit ASCII code for 7- or 9-track tape; odd parity.
'BCD'	Binary coded decimal for 7-track tapes; even parity.								
'BIN'	Unformatted binary for 7- or 9-track tapes; odd parity.								
'AS6'	6-bit ASCII code for 7-track tape; even parity.								
'ASC'	8-bit ASCII code for 7- or 9-track tape; odd parity.								
'VN=',vn	Two-digit version number (0 through 99). If VN=,vn is omitted, SIL enters '00' as the version number.								
'VSN=',vsn	Six-character volume serial number (VSN) of the requested tape. If VSN=,vsn is omitted or the specified VSN is zero, the operator assigns a tape to the file.								

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg is 80 bytes long.
'RFLUN=',rflun	Number SIL assigned to the file.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-35. Q5SETFIT Call Format (Sheet 3 of 3)

Call Format

CALL Q5SKIP(('LFN=',lfn
'FLUN=',rflun), optional parameters)

Calling Parameters

'LFN=',lfn	Name of a permanent file. LFN=,lfn must be specified if FLUN=,rflun is omitted.						
'FLUN=',rflun	Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.						
'COUNT=',cnt	Number of partitions to skip. If the number is a negative value, SIL backspaces the file the requested number of partitions. If COUNT=,cnt is omitted, SIL skips forward one partition.						
'PART=',part	Partition type to be skipped. If PART=,part is omitted, SIL skips records. <table border="0"> <tr> <td>'R'</td><td>Record.</td></tr> <tr> <td>'G'</td><td>Group.</td></tr> <tr> <td>'F'</td><td>File.</td></tr> </table>	'R'	Record.	'G'	Group.	'F'	File.
'R'	Record.						
'G'	Group.						
'F'	File.						

Return Parameters

'ERRLEN=',len	Error message length in bytes (integer).
'ERRMSG=',msg	Error message. The variable msg is 80 bytes long.
'STATUS=',stat	Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-36. Q5SKIP Call Format

Q5STATUS-GET FILE STATUS

The user calls the Q5STATUS call (refer to figure 9-37) to determine the status of a tape file. The file specified on a Q5STATUS call must not be a mass storage file.

Q5WRITE-WRITE BLOCK

The user calls the Q5WRITE routine (refer to figure 9-38) to transfer one or more blocks of data from a program buffer to a file. The user must attach the file and open it for explicit I/O before issuing the Q5WRITE call. The user can specify the buffer to be used on the Q5WRITE call; otherwise, SIL uses buffer 1 as specified in the FIT.

Unless the user specifies the WAIT parameter, SIL returns control to the caller immediately after issuing the request (before the data transfer is complete). The user should check for completion of the data transfer with a Q5CHECK call; however, if the user specifies the WAIT parameter, SIL does not return control to the caller until after the data transfer is complete.

If the user specifies a buffer on a Q5WRITE call with the BUFFER= parameter, the buffers specified in the FIT are no longer defined. Subsequent Q5READ and Q5WRITE calls must specify the BUFFER= parameter.

SIL does not check whether program buffers overlap.

For a magnetic tape file, only one I/O request can be pending at a time; therefore, if the file is on tape, the user must check that a Q5WRITE data transfer has completed before issuing the next Q5WRITE call specifying the file.

If, while writing data on a tape file, SIL senses the end of tape indicator, it performs one of the following actions according to the contents of the following fields in the FIT.

<u>vsn</u>	<u>stid</u>	<u>SIL Action</u>
Blank	Blank	Returns an end of tape status code.
Blank	Nonblank	Requests the operator to assign an available tape volume to the file.
Nonblank	Blank	Returns an end of tape status code.
Nonblank	Nonblank	Requests the operator to mount the next volume in the VSN list; if all volumes in the list have been written, SIL returns an end of tape status code.

Call Format

CALL Q5STATUS('LFN=',lfn
'FLUN=',rflun,optional parameters)

Calling Parameters

'LFN=',lfn Name of a tape file. LFN=,lfn must be specified if FLUN=,rflun is omitted.

'FLUN=',rflun Number SIL assigned to the file. FLUN=,rflun must be specified if LFN=,lfn is omitted.

Return Parameters

'DS=',ds Device status. SIL can return the following values.

B Busy.
N Not busy.
R Ready.

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg is 80 bytes long.

'LSTAT=',fp File position. SIL can return the following values.

BOI Beginning of information.
BOV Beginning of volume.
BOF Beginning of logical file.
MR Within a logical record.
EOR End of logical record.
EOG End of group (R and W files only).
EOV End of volume.
EOI End of information.

'OPS=',ops Operation status. SIL can return the following values.

DL Data lost.
EO End of operation.
PE Parity error.

Figure 9-37. Q5STATUS Call Format (Sheet 1 of 2)

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Return Parameters for Tapes Only

'DEN=',den Tape density. SIL can return the following ASCII values.

200	200 bpi
556	556 bpi
800	800 bpi
1600	1600 cpi

'POS=',pos Tape position. SIL can return the following ASCII values.

EOT	End of tape.
LP	Load point.

'WE=',we Indicates whether a write enable ring is inserted in the volume.

ON	Write enabled.
OFF	Write not enabled.

Figure 9-37. Q5STATUS Call Format (Sheet 2 of 2)

Call Format

CALL Q5WRITE({LFN=',lfn
'FLUN=',rflun'}, BYTCNT=',byt, optional parameters)

Calling Parameters

'LFN=',lfn Name of a permanent file. LFN=',lfn must be specified if FLUN=',rflun is omitted.

'FLUN=',rflun Number SIL assigned to the file. FLUN=',rflun must be specified if LFN=',lfn is omitted.

'BYTCNT=',byt Number of bytes SIL should write. The minimum unit SIL can write on mass storage is 1 block (4096 bytes). If the user specifies a byte count (for a mass storage file) that is not a multiple of 4096, SIL rounds the count up to the next multiple of 4096.

'BUFFER=',bfr Array to be used as the data buffer. The buffer must be on a page boundary (specified by a LOAD utility parameter). If the buffer is 128 blocks (a large page), it must be on a large page boundary. If BUFFER=',bfr is omitted and BUF2 is specified, SIL uses buffer number two; if BUFFER=',bfr and BUF2 are omitted, SIL uses buffer number one as specified in the FIT.

'BUF1' Indicates that SIL should use buffer number one as specified in the FIT. If BUF1 is omitted and either BUFFER=',bfr or BUF2 is specified, SIL uses the specified buffer. If BUFFER=',bfr, BUF1, and BUF2 are omitted, SIL uses buffer number one as specified in the FIT.

'BUF2' Indicates SIL should use buffer number two as specified in the FIT. If BUF2 is omitted and BUFFER=',bfr is specified, SIL uses the specified buffer. If BUF2 and BUFFER=',bfr are omitted, SIL uses buffer number one as specified in the FIT.

'WAIT' Indicates SIL should wait for completion of this write request before returning control to caller. If WAIT is omitted, SIL returns control immediately to the caller.

Return Parameters

'ERRLEN=',len Error message length in bytes (integer).

'ERRMSG=',msg Error message. The variable msg must be 80 bytes long.

'LEN=',rl Number of bytes transferred.

'RSN=',rsn Number assigned to the request. A Q5CHECK call uses this identifier.

'STATUS=',stat Status code. Refer to SIL Error Processing in section 8 for more information.

Figure 9-38. Q5WRITE Call Format

CHARACTER SET

A

The ASCII character set is shown in table A-1. Aids for hexadecimal-to-octal and hexadecimal-to-decimal conversion are given in tables A-2 and A-3.

TABLE A-1. AMERICAN NATIONAL STANDARD CODE FOR INFORMATION INTERCHANGE (ASCII)
WITH PUNCHED CARD CODES AND EBCDIC TRANSLATION

		0 0 0	0 0 1	0 0 1 0	0 0 1 1	0 0 1 0	0 0 1 1	0 0 1 0	0 0 1 1	1 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 0	1 1 1
b4 b3 b2 b1	COL ROW	0	1	2	3	4	5	6	7	8	9	10 (A)	11 (B)	12 (C)	13 (D)	14 (E)	15 (F)
0 0 0 0	0	NUL 12-0-9-8-1 NUL 00	DLE 12-11-9-8-1 DLE 10	SP no-punch SP 40	0 0 F0	@ S-4 @ 7C	P 11-7 P D7	8-1 79	p 12-11-7 p 97	11-0-9-8-1 DS 20	12-11-0-9-8-1 30	12-0-9-1 41	12-11-9-8 58	12-11-0-9-6 76	12-11-8-7 9F	12-11-0-8 88	12-11-9-8-4 DC
0 0 0 1	1	SOH 12-9-1 SOH 01	DC1 11-9-1 DC1 11	12-8-7 4F	1 1 F1	A 12-1 A C1	Q 11-8 Q D8	a 12-0-1 a 81	q 12-11-8 q 98	0-9-1 SOS 21	9-1 31	12-0-9-2 42	11-8-1 59	12-11-0-9-7 77	11-0-8-1 A0	12-11-0-9 B9	12-11-9-8-5 DD
0 0 1 0	2	STX 12-9-2 STX 02	DC2 11-9-2 DC2 12	8-7 7F	2 2 F2	B 12-2 B C2	R 11-9 R D9	b 12-0-2 b 82	r 12-11-9 r 99	0-9-2 FS 22	11-9-8-2 CC 1A	12-0-9-3 43	11-0-9-2 62	12-11-0-9-8 78	11-0-8-2 AA	12-11-0-8-2 BA	12-11-9-8-6 DE
0 0 1 1	3	ETX 12-9-3 ETX 03	DC3 11-9-3 DC3 13	# 8-3 # 7B	3 3 F3	C 12-3 C C3	S 0-2 S E2	c 12-0-3 c 83	s 12-11-3 s A2	0-9-3 23	9-3 33	12-0-9-4 44	11-0-9-3 63	12-0-8-1 80	11-0-8-3 AB	12-11-0-8-3 BB	12-11-9-8-7 DF
0 1 0 0	4	EOT 9-7 EOT 37	DC4 9-8-4 DC4 3C	\$ 11-8-3 \$ 58	4 4 F4	D 12-4 D C4	T 0-3 T E3	d 12-0-4 d 84	t 12-11-3 t A3	0-9-4 BYP 24	9-4 PN 34	12-0-9-5 45	11-0-9-4 64	12-0-8-2 8A	11-0-8-4 AC	12-11-0-8-4 BC	11-0-9-8-2 EA
0 1 0 1	5	ENQ 0-9-8-5 ENQ 2D	NAK 9-8-5 NAK 3D	% 0-8-4 % 6C	5 5 F5	E 12-5 E C5	U 0-4 U E4	e 12-0-5 e 85	u 12-11-4 u A4	11-9-5 NL 15	9-5 RS 35	12-0-9-6 46	11-0-9-5 65	12-0-8-3 8B	11-0-8-5 AD	12-11-0-8-5 BD	11-0-9-8-3 EB
0 1 1 0	6	ACK 0-9-8-6 ACK 2E	SYN 9-2 SYN 32	& 12 & 50	6 6 F6	F 12-6 F C6	V 0-5 V E5	f 12-0-6 f 86	v 12-11-5 v A5	12-9-6 LC 06	9-6 UC 36	12-0-9-7 47	11-0-9-6 66	12-0-8-4 8C	11-0-8-6 AE	12-11-0-8-6 BE	11-0-9-8-4 EC
0 1 1 1	7	BEL 0-9-8-7 BEL 2F	ETB 9-8-6 ETB 26	9-5 7D	7 7 F7	G 12-7 G C7	W 0-6 W E6	g 12-0-7 g 87	w 12-11-6 w A6	11-9-7 IL 17	12-9-8 GE 08	12-0-9-8 48	11-0-9-7 67	12-0-8-5 8D	11-0-8-7 AF	12-11-0-8-7 BF	11-0-9-8-5 ED
1 0 0 0	8	BS 11-9-6 BS 16	CAN 11-9-8 CAN 18	12-8-5 8 F8	8 8 F8	H 12-8 H C8	X 0-7 X E7	h 12-0-8 h 88	x 12-11-7 x A7	0-9-8 28	9-8 38	12-8-1 49	11-0-9-8 68	12-0-8-6 8E	12-11-0-8-1 B0	12-0-9-8-2 CA	11-0-9-8-6 EE
1 0 0 1	9	HT 12-9-5 HT 05	EM 11-9-8-1 EM 19	11-8-5 9 F9	9 9 F9	I 12-9 I C9	Y 0-8 Y E8	i 12-0-9 i 89	y 12-11-8 y A8	0-9-8-1 29	9-8-1 39	12-11-9-1 51	0-8-1 69	12-0-8-7 8F	12-11-0-1 B1	12-0-9-8-3 CB	11-0-9-8-7 EF
1 0 1 0	10 (A)	LF 0-9-5 LF 25	SUB 9-8-7 SUB 3F	11-8-4 8 F8	8-2 7A	J 11-1 J D1	Z 0-9 Z E9	j 12-11-1 j 91	z 12-11-9 z A9	0-9-8-2 SM 2A	9-8-2 3A	12-11-9-2 52	12-11-0 70	12-11-8-1 90	12-11-0-2 B2	12-0-9-8-4 CC	12-11-0-9-8-2 I(LVM) FA
1 0 1 1	11 (B)	VT 12-9-8-3 VT 0B	ESC 0-9-7 ESC 27	12-8-6 + 4E	11-8-6 9 F9	K 11-2 K D2	12-8-2 k 92	k 12-11-2 k 92	12-0 CO	0-9-8-3 CU2 2B	9-8-3 CU3 3B	12-11-9-3 53	12-11-0-9-1 71	12-11-8-2 9A	12-11-0-3 B3	12-0-9-8-5 CD	12-11-0-9-8-3 FB
1 1 0 0	12 (C)	FF 12-9-8-4 FF 0C	FS 11-9-8-4 IFS 1C	0-8-3 68	12-8-4 4C	L 11-3 L D3	12-11-3 l 93	l 12-11-3 l 93	12-11 i 93	0-9-8-4 2C	12-9-4 PF 04	12-11-9-4 54	12-11-0-9-2 72	12-11-8-3 9B	12-11-0-4 B4	12-0-9-8-6 CE	12-11-0-9-8-4 FC
1 1 0 1	13 (D)	CR 12-9-8-5 CR 0D	GS 11-9-8-5 IGS 1D	11 60	8-6 7E	M 11-4 M D4	12-11-4 m 94	m 12-11-4 m 94	11-0 DO	12-9-8-1 RLF 09	11-9-4 RES 14	12-11-9-5 55	12-11-0-9-3 73	12-11-8-4 9C	12-11-0-5 B5	12-0-9-8-7 CF	12-11-0-9-8-5 FD
1 1 1 0	14 (E)	SO 12-9-8-6 SO 0E	RS 11-9-8-6 IRS 1E	12-8-3 4B	0-8-6 6E	N 11-5 N D5	12-11-5 n 95	n 12-11-5 n 95	11-0 A1	12-9-8-2 SMM 0A	9-8-6 3E	12-11-9-6 56	12-11-0-9-4 74	12-11-8-5 9D	12-11-0-6 B6	12-11-9-8-2 DA	12-11-0-9-8-6 FE
1 1 1 1	15 (F)	SI 12-9-8-7 SI 0F	US 11-9-8-7 IUS 1F	0-1 61	0-8-7 6F	O 11-6 O D6	12-11-6 o 96	o 12-11-6 o 96	DEL 12-9-7 DEL 07	11-9-8-3 CU1 1B	11-0-9-1 E1	12-11-9-7 57	12-11-0-9-5 75	12-11-8-6 9E	12-11-0-7 B7	12-11-9-8-3 DB	EO 12-11-0-9-8-7 FF

LEGEND

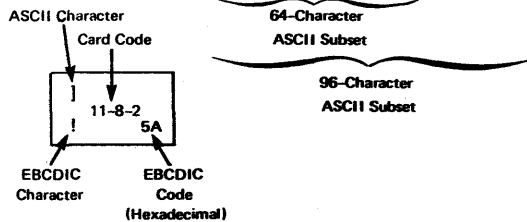



TABLE A-2. HEXADECIMAL-OCTAL CONVERSION

		First Hexadecimal Digit (Leftmost of a 2-digit number)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Second Hexadecimal Digit (Rightmost of a 2-digit number)	0	000	020	040	060	100	120	140	160	200	220	240	260	300	320	340	360
	1	001	021	041	061	101	121	141	161	201	221	241	261	301	321	341	361
	2	002	022	042	062	102	122	142	162	202	222	242	262	302	322	342	362
	3	003	023	043	063	103	123	143	163	203	223	243	263	303	323	343	363
	4	004	024	044	064	104	124	144	164	204	224	244	264	304	324	344	364
	5	005	025	045	065	105	125	145	165	205	225	245	265	305	325	345	365
	6	006	026	046	066	106	126	146	166	206	226	246	266	306	326	346	366
	7	007	027	047	067	107	127	147	167	207	227	247	267	307	327	347	367
	8	010	030	050	070	110	130	150	170	210	230	250	270	310	330	350	370
	9	011	031	051	071	111	131	151	171	211	231	251	271	311	331	351	371
	A	012	032	052	072	112	132	152	172	212	232	252	272	312	332	352	372
	B	013	033	053	073	113	133	153	173	213	233	253	273	313	333	353	373
	C	014	034	054	074	114	134	154	174	214	234	254	274	314	334	354	374
	D	015	035	055	075	115	135	155	175	215	235	255	275	315	335	355	375
	E	016	036	056	076	116	136	156	176	216	236	256	276	316	336	356	376
	F	017	037	057	077	117	137	157	177	217	237	257	277	317	337	357	377
Octal	000 -- 037	040 -- 077		100 -- 137		140 -- 177		200 -- 237		240 -- 277		300 -- 337		340 -- 377			

TABLE A-3. HEXADECIMAL-DECIMAL CONVERSION

		Exponent for Base 16					
		5	4	3	2	1	0
Hexadecimal Number	0	0	0	0	0	0	0
	1	1048576	65536	4096	256	16	1
	2	2097152	131072	8192	512	32	2
	3	3145728	196608	12288	768	48	3
	4	4194304	262144	16384	1024	64	4
	5	5242880	327680	20480	1280	80	5
	6	6291456	393216	24576	1536	96	6
	7	7340032	458752	28672	1792	112	7
	8	8388608	524288	32768	2048	128	8
	9	9437184	589824	36864	2304	144	9
	A	10485760	655360	40960	2560	160	10
	B	11534336	720896	45056	2816	176	11
	C	12582912	786432	49152	3072	192	12
	D	13631488	851968	53248	3328	208	13
	E	14680064	917504	57344	3584	224	14
	F	15728640	983040	61440	3840	240	15



$j_{16} \times 16^i = m_{10}$

To find $E_{16} \times 16^3$; look at row E, column 3 and find 57344

Table B-1 lists the messages produced by the system utilities described in this manual. USER1 errors detected by CARDREDR are printed in the user dayfile and are not returned to the operator's terminal. The return codes are compared to the threshold value specified by a TV control statement in a batch job.

Code interpretation is:

- 0 No error
- 4 Warning (nonfatal) errors
- 8 Fatal error and task abort

Table B-2 lists SIL error messages. Each message is 80 characters long and has the following format.

The severity of the routine code description follows.

severity	Error severity, either fatal or warning. If the user does not specify the STATUS= parameter on the call, SIL returns the appropriate return code to the program's controller.
routine	Name of the SIL routine that detected the error. Table B-2 lists the routines that can issue the error.
code	Status code. The messages are listed in order by status code.
description	Description of the error.

TABLE B-1. DIAGNOSTIC MESSAGES

Message	Significance/Action	Return Code	Issued By
ADDR=REGnn	Address indicated by the symbol specified corresponds to register nn.	4	DEBUG
ADR IN REG FILE	A location, or the computation of name=location, was less than #4000.	4	DEBUG
ADR NOT COD ADR	Address specified in EXECUTE, BKPT, or BKPTR command is not that of executable code. No action will be taken.	4	DEBUG
ADR BEYOND MOD	Address specified does not fall within the current module.	4	DEBUG
ALL DONE	CYBER 200 OS has completed the task.	-	Op System
ALPHA OUT OF BOUNDS	Address of Alpha message is out-of-bounds.	-	Op System
ATTEMPT TO EXTEND PAST MAX ON FILE	Original length of file too small. Rerun with a larger file.	-	Op System
ATTEMPT TO READ PAST EOF ON FILE	Do not attempt to read past end of file.	-	Op System
ATTEMPTED TO CALL NON-COMDECK	Correct CALL statement	4	Update
ATTEMPTED TO READ FROM UNKNOWN FILE	Correct READ statement.	8	Update
BAD ACCOUNT	Invalid account on a LOGON statement.	-	Op System
BAD BINARY FILE Ifn THE WORD 'MODULE' NOT FOUND	Self-explanatory.	8	LOAD
BAD CARD ENCOUNTERED	Correct input file card.	4	Update
BAD COMMAND	Correct DEBUG directive.	4	DEBUG
BAD DFM IN CTEE	The controllee to be debugged had been loaded with only a portion of the Data Flag Manager. When the controllee contains DFM, DEBUG links to the controllee's copy of DFM. When the controllee does not contain DFM, DEBUG uses its own copy. However, when the controllee contains only part of the DFM, DEBUG assumes something is wrong and aborts. The controllee should contain all or none of the following module/entry points: FT_SYSTEM/Q7DFINIT; DFBH_/Q7DFCL1.	8	DEBUG
BAD FUNCTION CODE FOR GET/SEND MESSAGE	Notify a systems analyst.	8	Update
BAD JOB FILENAME AND/OR ACCOUNT NUMBER (FUNCTION=#000B, R=xx)	A batch input file must have its account number changed to the one specified on its job card. The error occurred on a function #000B call (Change File Attributes).	-	CARDREDR CYBERIN
BAD LEVEL	Invalid security level.	-	Op System
BAD LIB FILE Ifn THE WORD 'LIBRARY' NOT FOUND	Self-explanatory.	8	LOAD
BAD LOGON ERR	LOGON statement is in error.	-	Op System
BAD MINUS PAGE	Minus page of the task is not set up correctly.	-	Op System

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
BAD NAME NO NAME OR DUPLICATE IDENT	Correct Update directive.	4	Update
BAD ORIGIN ADDRESS FOR GROUP	An origin address must be on a small or large page boundary.	8	LOAD
BAD SUFFIX	Valid values are A, B, C, or D.	-	Op System
BAD USER/ACCOUNT	Invalid user number and account on a LOGON command.	-	Op System
BAD USER/SUFFIX	Invalid user number and account on a LOGON command.	-	Op System
BATCH PROCESSING TASK NOT INITIATED BECAUSE - reason	Self-explanatory.	-	CARDREDR
BATCH PROCESSOR RUNNING ON THAT SUFFIX	User tried to logon with a batch suffix.	-	Op System
BKPT NOT FOUND	Correct BKPTR directive.	4	DEBUG
BKPT TABLE FULL	No additional breakpoints can be set until one or more existing breakpoints are removed.	4	DEBUG
BOUND IMPLICIT MAP ANOMALY	Garbage in the minus page. Rebuild the minus page.	-	Op System
CALL OF INACTIVE OR PURGED COMDECK	Informative message.	4	Update
CALL OF NON COMDECK	Change CALL to existing common deck name.	4	Update
CALLER IS NOT A PRIVILEGED USER	Program has detected that the caller is attempting to illegally process files.	8	PF
CANNOT OPEN ifn	Self-explanatory.	8	DUMP
CANNOT RESTART - REGTBL full	Cannot restart a drop file. System table REGTBL is full. Try again later.	-	Op System
CANT DESTROY EXISTING DROP FILE	Modified pages written to drop file and cannot be purged.	-	Op System
CENTRAL MEMORY PARITY ERROR	Rerun job.	-	Op System
COMDECK ARRAY TOO SMALL	Too many comdecks for the comdeck array.	8	Update
reason - COMMAND IGNORED	When the parameters for a LOOK directive are meaningless, missing, or illegal, the directive is ignored. Refer to the reason given for more information.	4	LOOK
COMPARE TERMINATED - END OF FILE ifn	Informative message.	-	COMPARE
CONTRADICTION, LIST=0 AND OUTPUT=	Correct control statement.	8	OLE
CONTROL CARD ERROR	Correct control statement syntax. Parameters should be separated from the control statement name by a (, / = + - or blank.	0	BATCHPRO
CONTROLLEE FORMAT ERROR	Error in format of the controllee option.	8	LOAD
CONTROLLEE FILE ifn IS TOO SMALL, NEED nnnnnn SMALL PAGES	Rerun with a larger controllee file specified.	8	LOAD
COULD NOT CREATE NEWPL	No mass storage space available. Rerun job.	8	Update

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
COULD NOT LINK CORRECTION TO DIRECTORY ENTRY	Notify a systems analyst.	4	Update
CR STR TOO BIG	Character string is bigger than space allotted; too many characters in the input symbol.	8	LOAD
CREATE PERMANENT FILE Ifn	Informative message.	-	DEFINE
CREATED PERMANENT FILE	The system successfully created a permanent file.	-	DEFINE
CREATION RUN ABORT	Informative message.	8	Update
CREATE PERMANENT FILE-LFN	Informative message	0	DEFINE
DATA BASE EXCEEDS FIELD	Length of the data base is greater than #FFFFFFF words.	8	LOAD
DATA BEYOND FF	The user has done an EREG such that the data to be entered would go beyond entering in register #FF. For example, EREG FE,1,2,3,4,5.	4	DEBUG
DECK DOES NOT EXIST	Change deck name to that of an existing deck.	4	Update
DECKS OR IDENTs OUT OF ORDER	Correct range parameter.	4	Update
DEBUG ERROR. TRY AGAIN	DEBUG failed when attempting to build its internal tables. The error could be the result of a temporary condition; the user should resubmit the job.	8	DEBUG
DROP FILE IOC DOES NOT VERIFY	Either the drop file does not exist or the ioc does not match the file index.	-	Op System
DROP FILE IS PRI-OPENED	Cannot restart a drop file that is open. It might be open due to a privileged DUMPF.	-	Op System
DROP FILE MAP FULL, PAGE NOT MAPPED	Reprogram job to use fewer drop file map entries at one time.	-	Op System
DROP FILE OVERFLOW	Rerun with a larger drop file.	-	Op System
DROP FILE OVERFLOW CAUSED BY CALL TO VS	Rerun with a larger drop file.	-	Op System
DROP FILE TOO SMALL	New drop file will not hold existing drop file page.	-	Op System
DUMPING Ifn	Informative message identifying file being dumped.	-	DUMPF
DUPLICATE PUBLIC FILES	Informative message at the operator's console during LOGON.	-	Op System
EMPTY INPUT FILE	Correct input file.	8	Update
EMPTY INPUT FILE IN Q MODE	Q mode requires an input file.	8	Update
ENCOUNTERED INVALID CHARACTER	Correct input card.	4	Update
ENCOUNTERED READ ERRORS ON OLDPL (BAD DATA)	Notify a systems analyst.	8	Update

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
ENCOUNTERED UNPROCESSED MODIFICATIONS	Correct Update directive.	4	Update
ERROR IN ATTEMPT TO DEQUEUE SERVICE STATION FILE	Notify a systems analyst.	-	CARDREDR
ERROR IN ATTEMPT TO TRANSFER CARD INPUT FILE TO MASS STORAGE	Notify a systems analyst.	-	CARDREDR
ERROR IN CREATING CENTRAL FILE FOR CARD INPUT FILE AND ERROR IN ATTEMPT TO DESTROY SERVICE STATION FILE (FUNCTION = #0001, SS = #ss)	See system message CREATE FILE for explanation of error code.	-	CARDREDR
ERROR IN Q7PROMPT	Notify a systems analyst.	8	OLE
ERROR OCCURRED IN ATTEMPT TO DESTROY INPUT FILE ON SERVICE STATION DEVICE	Notify a systems analyst.	-	CARDREDR
ERROR OCCURRED IN ATTEMPT TO READ INPUT FILE FROM SERVICE STATION DEVICE	Notify a systems analyst.	-	CARDREDR
ERROR OPENING FILE Ifn	Q5OPEN failed to open file Ifn.	8	LOAD
ERROR OPENING LIBRARY FILE Ifn	Q5OPEN failed to open file Ifn.	8	LOAD
EXCEEDED MAXIMUM SEQUENCE NUMBER	Maximum sequence number is 65 535.	4	Update
EXCEEDED SPECIFIED PN'S	List of pack names insufficient for file creation. Processing continues with system default PN.	-	PF
EXISTING PERMANENT FILE Ifn OPENED	File Ifn exists as an attached permanent file (informative message).	-	Q5GETFIL
EXISTING LOCAL FILE Ifn MADE PERMANENT	Informative message.	-	DEFINE
EXPECTED/BAD NAME ENCOUNTERED	Notify a systems analyst.	4	Update
EXPECTED FILE NAME NOT ENCOUNTERED	Correct Update directive.	4	Update
EXPECTED IDENT NAME NOT FOUND	Correct Update directive.	4	Update
EXPECTED SEQUENCE NUMBER NOT FOUND	Correct Update directive.	4	Update
FATAL SYSTEM ERROR	Rerun job.	-	Op System
FILE INDEX FULL. NONE OF YOUR PRIVATE FILES AVAILABLE.	Other users must destroy some of their files or logoff to free space in the file index. Relogon (reenter the LOGON command after having previously issued a BYE) to bring in private files.	-	Op System

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
FILE Ifn DOES NOT EXIST	No file assigned to the job has the name specified on the ROUTE statement.	8	ROUTE
FILE Ifn GIVEN TO POOL pool	The system successfully gave the specified file to the specified pool.	-	GIVE
FILE SEGMENT TABLE IS FULL	Rerun job.	-	Op System
FILES COMPARED EQUALLY	Informative message.	0	COMPARE
FIRST FILENAME ILLEGAL - NO FILES PURGED	Self-explanatory.	8	PURGE
FOLLOWING CARDS ARE SKIPPED - NOT IN INSERT MODE	Informative message.	4	Update
FORMAT ERROR	Correct control statement.	8	DEBUG EDITPUB
GENERATION OF UNIQUE IDNAME FAILED	Notify a systems analyst.	4	UPDATE
GROUP ORIGIN AT ADDRESS WHICH IS ALREADY ALLOCATED	An attempt was made to allocate a group of modules or common blocks where another module or common block is already allocated.	8	LOAD
IDENT DOES NOT EXIST	Correct Update directive.	4	Update
ILLEGAL BKPT	DEBUG has obtained control from an unexpected point in the controllee, namely, from a point at which DEBUG has not set a breakpoint.	4	DEBUG
ILLEGAL CHAR NUMBER	Correct hexadecimal number to include only digits 0 through 9 and letters A through F.	8	LOAD
ILLEGAL COMMAND	Illegal input parameters.	8	LOAD
ILLEGAL C504 REQUEST	User jobs cannot issue a C504 request.	-	Op System
ILLEGAL DATE CORRECTED TO mmddyy	DT parameter is not a legal date; it is changed to the nearest legal date as shown.	-	PF
ILLEGAL FIRST CARD	STORE card required.	-	BATCHPRO
ILLEGAL INSTRUCTION	Illegal instruction at address given.	-	Op System
ILLEGAL PARAMETER	Self-explanatory.	8	OLE
ILLEGAL RECORD TYPE FOR FILE outfil	The output file outfil has record type F or U. The user should rerun the task using either another output file or no output file.	8	DEBUG OLE
ILLEGAL REQUEST	Illegal Alpha function code.	-	Op System
INADR EXCEEDS infile FILE LENGTH	I parameter is greater than the number of words in infile.	8	COPY
INT DATA ** MODE = 1 TYPE ILLEGAL	Mode 1 of an interpretive data type is illegal - probably the most common with type 9 mode 1. Table type 101.	8	LOAD
INT REL ** MODE = 2 TYPE DOES NOT EXIST	Mode 2 of an interpretive data type does not exist. Table type 201.	8	LOAD
INTERACTIVE ACCESS NOT ALLOWED AT THIS TIME	The operator has turned off the INTRACTV job category, preventing interactive access to the system.	8	XEQLN
INTERNAL ERROR IN UPDATE, NO. x	Consult a systems analyst.	8	Update

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
INVALID CARD	Correct input.	4	Update
INVALID DIRECTIVE FOR CREATION RUN	Only READ, DECK, COMDECK, and ADDFILE can appear in a creation run.	4	Update
INVALID EXPRESSION: "expression" ENTER REPLACEMENT OR CANCEL	Enter correct expression, enter CANCEL to abort, or carriage return to ignore bad expression.	8	Q7KEYWRD
INVALID FILE NAME	File names must be one through eight letters or digits beginning with a letter.	4	Update
INVALID FILE OR FUNCTION	Correct Update input.	8	Update
INVALID FILE NUMBER OR FUNCTION	Correct Update input.	4	Update
INVALID SEQUENCE NUMBER	Correct Update directive.	4	Update
INVALID TIME LIMIT	The user specified an incorrect time limit on the RESOURCE statement or on the execute line. The time limit must be a decimal integer between 0 and 599 940.	8	IQM XEQLN
INVALID UPDATE CHECK WORD	Notify a systems analyst.	8	Update
INVALID USER NUMBER	User tried to use a reserved number.	-	Op System
I/O ERROR RECEIVED BY WRPLY	Rerun job.	-	Op System
IOC DOESN'T VERIFY	Invalid IOCs in a drop file being restarted. Usually means there is an IOC for a file which is no longer available.	-	Op System
IOC FOR Ifn ALREADY IN USE	Self-explanatory.	8	DEBUG
JOB ABORTED	Informative message.	-	CARDREDR
JOB CATEGORY SPECIFIED DOES NOT EXIST	The input queue manager does not recognize the job category mnemonic specified on the RESOURCE statement. Ask installation personnel for a valid mnemonic or specify JDEFAULT.	8	IQM
JOB FILE VACUOUS	The first record of a batch job must contain ASCII character control statements.	-	BATCHPRO
KILL	Operator response to tape condition. Tape operation is aborted.	8	Tape Sub
LARGE PAGE LIMIT EXCEEDED	An instruction in the program requires more large pages than the current large page limit. Increase the large page limit.	-	Op System
LARGE PAGE LIMIT EXCEEDS MAX WORKING SET	The specified large page limit exceeds the maximum working set for the job. Specify a smaller large page limit on the RESOURCE statement.	8	IQM
LARGE PAGE LIMIT OF nnn PAGES EXCEEDED	The user specified a large page limit on the RESOURCE statement that exceeds the maximum memory available in the machine. Specify a large page limit not greater than nnn blocks.	8	IQM
LENGTHS DON'T MATCH FOR COMMON BLOCK BLOCKNAME	Self-explanatory.	4	LOAD
LIBRARY DIRECTORY AND INDEX TABLE FULL	Notify a system analyst.	8	OLE

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
LINK I/O ERROR (FC)=XX	Error in explicit I/O other than EOF or operator abort. Contents of register FC are given.	8	SAVEPF
LINKED MAINFRAME FILE ERROR	File error on CYBER or aborted by CYBER 200 operator. Check parameters and try again. (CYBER 200 link station only.)	8	GETPF SAVEPF PURGE
LOADING Ifn	Informative message identifying file being loaded.	-	LOADPF
LOADMAP FORMAT ERROR	Bad input parameter for creating an output file.	8	LOAD
MASTER CONTROL WORD DOES NOT MATCH OLD PL	Notify a systems analyst.	4	Update
MASTER DIRECTORY FILE xxxxxxxx IS FULL	The master directory, xxxxxxxx, containing the pseudo file names for the user's dumped files on disk is full. Remaining files are dumped to the next specified PN (packid).	4	PF
MAX WORKING SET LIMIT OF nnn BLOCKS EXCEEDED	The user specified a working set size limit on the RESOURCE statement that exceeds the maximum memory available in the machine. Specify a working set size limit not greater than nnn blocks.	8	IQM
MAXIMUM ERRORS EXCEEDED	More than 256 errors. Correct and rerun.	4	Update
MAXIMUM NUMBER OF CORRECTION HISTORY BYTES REACHED	Create new program library.	4	Update
MAXIMUM NUMBER OF FIELDS EXCEEDED	Correct control statement.	4	Update
MESSAGE ERROR	Error in system GET A MESSAGE call.	8	LOAD
MISSING INPUT FILE	Correct control statement.	8	OLE
MODULE LIMIT ON OBJECT FILE EXCEEDED	Notify a systems analyst.	8	OLE
MODULE ON FILE Ifn HAD A MODULE HEADER LENGTH OF ZERO	Bad module header format in the input file.	8	OLE
MODULE ON FILE Ifn HAS NO HEADER	Module does not have a header table.	8	OLE
MODULES, NAMED, AND BLANK COMMON CANNOT BE GROUPED TOGETHER	The user specified blocks of more than one type (module, named common, and blank common) with a grouping parameter. Each type must be specified with a separate parameter.	8	LOAD
MORE THAN ONE ALTERNATE FILE WAS ATTEMPTED	Correct Update directive.	8	Update
MORE THAN ONE OMIT FOR FILE Ifn	Correct control statement.	8	OLE
MTuu message	See corresponding NTuu message.		
NAME ALREADY EXISTS IN DIRECTORY	Change duplicate deck name.	4	Update
NAME DOES NOT EXIST IN DIRECTORY	Change deck name or directive requested.	4	Update

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
NO ALPHA POINTER	Virtual system was called for an Alpha message, but the pointer to the Alpha message was 0.	-	Op System
NO DISC SPACE FOR EXTENSION ON FILE	Rerun job with this file on a disk that has enough space for this file.	-	Op System
NO DROP FILE	There is no drop file for this task.	-	Op System
NO ENTRY	No entry point found.	8	LOAD
NO ERROR EXIT ADDRESS	Task issued an Alpha/Beta system call which returned an error, but the error exit address field in the Alpha portion was 0.	-	Op System
NO EXT/ENT POINTER FILE lfn	No type 2 table pointer found.	8	LOAD
NO FILE	Either a file with the specified name does not exist, or the number of characters in filename was not 1 through 8.	-	Op System
NO FILE NAME FOR READ	Add file name to READ directive.	4	Update
NO FILE SEGMENT TABLE SPACE	System file segment table full. Rerun.	-	Op System
NO FILES TO LIST	No files exist that match the specifications on the FILES control statement.	0	FILES
NO FILES TO xxxxxx	Informative message.	0	PF
NO FST ORDINAL WITH C50X REQUEST	Rerun with an FST ordinal.	-	Op System
NO LIBRARY DIRECTORY SEARCH	Job has an address in library space. Library space cannot be used at this time.	-	Op System
NO LOGONS	Interactive terminal logon lines are inhibited.	-	Op System
NO MESSAGE POINTER FOLLOWS EXIT FORCE	Rerun with a correct exit force.	-	Op System
NO MORE SEGMENT SPACE IN FILE1, FILE lfn	File has been extended three times. Rerun with a larger file.	-	Op System
NO PARAMETERS SPECIFIED	Correct DEBUG control statement.	8	DEBUG
NO PP	No task in execution to break.	-	Op System
NO SOURCE FILE	There is no controllee file.	-	Op System
NO SUCH MODULE	Value entered for name=location was not the name of a module in the controllee.	4	DEBUG
NO SUCH SYMBOL	Symbol specified was not found as a symbol of current type (S or L) in current module.	4	DEBUG
NO SWITCH PARAMETERS FOUND, FILE NOT ALTERED.	File name was only parameter found.	4	SWITCH
NO TIME IN BANK	Time in repository bank is reduced to zero.	-	Op System
NO TIME LEFT FOR drop file lfn	Rerun with a larger time limit.	-	Op System
NO TL	Zero time limit (TL) specified.	-	Op System
NO WRITE PERMISSION - COMMAND IGNORED	Self-explanatory.	-	LOOK

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
NON-EXECUTABLE FILE	File requested is not a virtual code file (file type other than 2).	-	Op System
NON-MATCHING WORDS	Listed words did not compare equally.	4	COMPARE
NOT A LOGON	First word of the LOGON command must be LOGON.	-	Op System
NOT A USER	Invalid user number.	-	Op System
NOT ENOUGH TIME FOR THIS JOB	Time limit specified exceeds time remaining in repository bank.	-	Op System
NOT EXPECTING SEQUENCE NUMBER	Correct Update directive.	4	Update
OLDPL DIRECTORY CANNOT BE PROCESSED	Notify a systems analyst.	8	Update
OFFSET NOT ALLOWED FOR COPY TO TAPE	The user specified an offset value using the O= parameter when copying to a tape file.	8	COPY
OLE TERMINATED	Informative message.	-	OLE
OMIT FILE ifn NOT AN INPUT FILE	Correct control statement.	8	OLE
OMIT MODULE modname NOT ON FILE ifn	The module does not exist in the input file.	8	OLE
OMIT PARAMETER MISSING A FILE OR MODULE	Correct control statement.	8	OLE
ON TTY xxxx	User numbers cannot logon to more than one terminal at a time.	-	Op System
OPERATOR NO.=uuuuuu	The system operator attempted to logon under an invalid user number. The correct operator number is uuuuuu.	-	Op System
ORIGIN NOT ON LARGE PAGE BOUNDARY - ORIGIN = neworigin	Warning - GRLPALL option.	4	LOAD
OUT OF BOUND MEMORY REFERENCE	Attempt to access space outside the task virtual space.	-	Op System
OVERDRAWN	Insufficient time remains in repository bank; results from G status request.	-	Op System
OUTADR EXCEEDS outfile FILE LENGTH	O parameter is greater than the number of words in outfile.	8	COPY
packid PACK NOT AVAILABLE	Specified disk pack was not available; the program continues with the next specified pack identifier PN (packid).	4	PF
PAGE SIZE CONFLICT IN DROP FILE	Small page fault but the drop file is in large pages, or vice versa.	-	Op System
PARAMETER OR FORMAT ERROR	Error in parameter specifications.	8	Pool Utilities PF
PARAMETER OR FORMAT ERROR FOUND. UTILITY TERMINATED	Correct FILES control statement.	8	FILES
PARAMETER OR FORMAT ERROR FOUND. UTILITY TERMINATED	Correct GIVE control statement. File remains with old owner.	8	GIVE

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
PC AND RMK MUST BE SINGLE CHARACTER OR HEX NUMBER FROM 0 TO FF	The padding character and/or the record mark character must be specified as the character or its hexadecimal code.	8	DEFINE REQUEST SWITCH
POOL pool ATTACHED	The system successfully attached the specified pool.	-	PATTACH
POOL pool CREATED	The system successfully created the specified pool.	-	PCREATE
POOL pool DESTROYED	The system successfully destroyed the specified pool.	-	PDESTROY
POOL pool DETACHED	The system successfully detached the specified pool.	-	PDETACH
POOL FILE lfn CURRENTLY OPENED	Self-explanatory.	4	PURGE
PROBLEM WITH FILE	Notify a systems analyst.	8	Update
PROCESSED INVALID DIRECTIVE IN ALTERNATE FILE	Correct directive.	4	Update
PURGE ERROR R=#rrr SS=#ss ON FILE lfn	See DESTROY system message in volume 2 for an explanation of the error code.	8	PURGE
Q7KEYWRD ERROR - RESPONSE = ss	The keyword checking routine found an error in the parameters specified on the ROUTE control statement.	8	ROUTE
RANGE IS NOT PERMITTED FOR YANK DECK	Correct YANKDECK directive.	4	Update
READ ERROR ON FILE	SIL detected an error. Refer to the SIL message.	8	Update
RECURSIVE CALL NOT PERMITTED	Correct CALL references.	4	Update
REPEAT OPTION IGNORED - NOT VALID WHEN DEFERRED ROUTE SELECTED	The user cannot specify both the REP and DEF parameters on a ROUTE control statement.	8	ROUTE
REQUESTED LP EXCEEDS INTRACTV LP LIMIT OF nnn PAGES	The large page limit specified on the execute line exceeds the maximum large page limit for the INTRACTV job category. Reenter the execute line specifying a large page limit not greater than nn pages.	8	XEQLN
REQUESTED LP EXCEEDS JOB'S LP LIMIT OF nn PAGES	The user specified a large page limit that exceeds the maximum large page limit for the job. Correct the SET statement so the large page limit is not greater than nnn pages.	8	BATCHPRO
REQUESTED LP EXCEEDS TASK WS LIMIT OF nnnn BLOCKS	When multiplied by 128, the specified large page limit exceeds the working set size limit of nnnn blocks. Reenter the execute line specifying a larger working set size limit or a smaller large page limit.	8	XEQLN
REQUESTED WS EXCEEDS INTRACTV WS LIMIT OF nnnn BLOCKS	The user specified a working set size limit larger than the maximum working set size limit for the INTRACTV job category. Reenter the execute line specifying a working set size limit not greater than nnnn blocks.	8	XEQLN

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
REQUESTED WS EXCEEDS JOB'S WS LIMIT OF nnnn BLOCKS	The user specified a working set size limit that exceeds the maximum working set size limit for the job. Correct the SET statement so the working set size limit is not greater than nnnn blocks.	8	BATCHPRO
REQUESTED WS TOO SMALL FOR JOB'S LP LIMIT OF nnn PAGES	The user specified a working set limit smaller than the current large page field length. Correct the SET statement so the working set size limit is not smaller than nnn*128.	8	BATCHPRO
REQUIRED PARAMETER MISSING. NEXT EXPRESSION IS: expression ENTER PARAMETER OR "CANCEL"	Required positional parameter missing. Given expression appeared in the position where a required parameter was expected.	8	Q7KEYWRD
SAY AGAIN	Special character (sc) is not followed by valid system inquiry character.	-	Op System
SBU MEMORY PARITY ERROR	Parity error occurred on read or write.	-	Op System
SECURITY LEVEL TOO HIGH	Invalid security level for this user number.	-	Op System
SEND AGAIN	The state at this DB entry is zero; or job class is priority and privileged job permission flag is zero; or job is currently in interrupt mode, explicit I/O interrupt has occurred.	-	Op System
SEQUENCE NUMBER EXCEEDED	Create two decks if text cards exceed 65 535.	4	Update
SEQUENCE NUMBER MISSING FROM DIRECTIVE	Add a sequence number to the directive.	4	Update
SEQUENCE NUMBER NOT FOUND	Specified sequence number nonexistent. Correct.	4	Update
SOURCE OR DROP FILE ANOMALY	IOC in bound implicit map is not 16 (source); or bound implicit map entry is outside of file bounds; or drop file (free space) map address overlap occurred.	-	Op System
SYNTAX ERROR ON DIRECTIVE	Correct error and resubmit.	4	Update
SYSTEM DROP FILE CREATE ERROR	Either the disk or file index is full.	-	Op System
SYSTEM MESSAGE ERROR	Batch processor detected a system message error. Contents of Alpha and Beta words follow.	-	BATCHPRO
SYSTEM TABLES FULL, TRY AGAIN	The XEQ buffer table is full as more than 8 execute lines have been entered; or no DB entry can be obtained; or no user table entries are available.	-	Op System
TABLE OVERFLOW	The loaded modules contain too many external or common references to list in a cross-reference list. Rerun the job, omitting the LO=x parameter from the LOAD statement.	8	LOAD
TABLE TYPE NOT IMPLEMENTED	Refers to compiler output for the loader.	8	LOAD
TASK SUSPENDED WAITING FOR PERMANENT FILE	A file specified on the ATTACH statement is already attached to another suffix. The user specified the WAIT parameter so the task is suspended until the file is available.	0	ATTACH

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
THERE EXISTS MORE FILES THAN CAN BE PROCESSED	In issuing the system message LIST FILE INDEX for private user files, the area to contain the file entries is not large enough. Subsequently, DUMPF processes only the files received in the area. In order to obtain the remaining files, the number of files must be reduced (for example: P option or destroy some files).	0	PF
TOO MANY ERRORS	Correct Update directives.	8	Update
TRANSMISSION PARITY ERROR	Parity error occurred on read or write.	-	Op System
TRY AGAIN	ROLL or BACK was given before a DISPLAY or ENTRY command and DEBUG has no point of reference for ROLL/BACK.	4	DEBUG
	CONTINUE was given before EXECUTE. A second EXECUTE command is given. STEP is given before EXECUTE.		
UNABLE TO CLOSE SERVICE STATION INPUT FILE	Notify a systems analyst.	-	CARDREDR
UNABLE TO DESTROY POOL	Users attached, files are in the pool, or user is not the pool boss.	4	PDESTROY
UNABLE TO ENLARGE DROP FILE. TRY AGAIN	Self-explanatory.	8	PURGE
UNABLE TO FIND EXECUTE FILE	Self-explanatory.	-	Op System
UNABLE TO FIND EXTERNAL-ENTRY TABLE FOR MODULE modname ON FILE lfn	OLE could not find the external reference table due to bad module structure.	8	OLE
UNABLE TO GET TIME AND DATE	Notify a systems analyst.	8	OLE
UNABLE TO OPEN FILE LOCATED ON SERVICE STATION DEVICE	Self-explanatory.	-	CARDREDR
UNABLE TO SAVE FILE	Error while copying file. ROUTE aborted.	8	ROUTE
UNABLE TO UNLOCK CORE PREVIOUSLY LOCKED DOWN FOR I/O FILE TRANSFER	Self-explanatory.	-	CARDREDR
UNDEFINED NAME OR ALREADY IN GROUP	Grouping not done because of an undefined name or the element to be grouped is already in another group.	8	LOAD
UPDATE OBTAINED BAD DATA IN PROCESSING THIS CARD	Correct card.	4	Update
USER LOCKED OUT OF SPECIFIED JOB CATEGORY	The user is not validated to use the job category specified on the RESOURCE statement. Specify another mnemonic such as the default, JDEFAULT.	8	IQM
VARIABLE RATES NOT DEFINED AT THIS INSTALLATION	Control statement contained VRI parameter and system installation parameter IP_F_VR is set to zero.	8	EDITPUB
WARNING *** ATTACHED POOLS	Informative message during LOGON. The user has attached pools.	-	Op System
WARNING - ATTRIBUTES OF POOL FILE MAY NOT MATCH INPUT FILE	The COPY utility cannot change pool file attributes.	4	COPY

TABLE B-1. DIAGNOSTIC MESSAGES (Contd)

Message	Significance/Action	Return Code	Issued By
WARNING *** CHECKPOINTED JOBS UNDER SUFFIX D jobname 1, jobname 2, jobname 3, jobname 4, jobname 5,	A logon under the checkpointed batch suffix D will abort the checkpointed jobs for this user. All checkpointed batch input files, checkpointed output files, and batch local files are destroyed. Permanent files attached to the checkpointed suffix are returned.	-	Op System
WARNING * DUPLICATE FILES	Informative message during LOGON. The user has duplicate files.	-	Op System
WARNING MODULE name FROM FILE lfn IS INACCESSIBLE AND THEREFORE NOT LOADED	Module is deleted from controllee since it cannot be referenced.	4	LOAD
WARNING MULTIPLE TRANSFER SYMBOLS DEFINED	More than one program entry point is defined.	4	LOAD
WARNING - NO OBJECT FILE CREATED	Informative message.	4	OLE
WARNING - OBJECT FILE LENGTH INCREASED TO nnnn PAGES	Informative message.	4	OLE
WARNING - PACKID IGNORED. filenam ON PACK packid	File filenam is not on the pack specified on the COPY statement. It is on pack packid and the file was copied to that pack.	4	COPY
WARNING UNSATISFIED EXTERNAL(S) DETECTED DURING LOAD	Routine referenced but not provided.	4	LOAD
WARNING xxxxxxxx IS DUPLICATE DUPLICATE ENTRY POINT IN MODULES yyyyyyyy AND zzzzzzzz	xxxxxxx is the entry name, and yyyyyyyy and zzzzzzzz are the module names. References to xxxxxxxx link to the entry in yyyyyyyy, which was the first encountered.	4	LOAD
taskname WORKING SET TOO SMALL nn% OF CPU TIME	The maximum working set size for task taskname was too small for nn% of its execution time. The user should consider increasing the maximum working set size limit to decrease the paging required for the task.	4	BATCHPRO
WRITE ON READ-ONLY FILE	Job attempted to write to temporary space for which the job does not have write access.	-	Op System
WRITE VIOLATION IN SYSTEM CALL	Operating system error on read-only file.	-	Op System

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES

Error Code	Message	Significance	Issuing Routine
F 0001 to 0199	ILLEGAL PARAMETER parameter	The user specified an invalid or incorrect parameter. The status code in the message is the ordinal of the parameter within the parameter sequence.	ALL
F 0200	SIL BUG-ILLEGAL OPTION	SIL specified an illegal option for a system message. Consult a systems analyst.	ALL
F 0201	SIL BUG-ILLEGAL BETA	The Beta area SIL supplied for the system message was rejected by the system. Consult a systems analyst.	ALL
F 0202	SIL BUG-UNRECOGNIZED ERROR CODE	Unrecognized R code in an Alpha or an unrecognized SS code in a Beta. Consult a systems analyst.	ALL
F 0203	SIL BUG - INTERNAL CALL TO routine FAILED	Internal error. Consult a systems analyst.	ALL
F 0204	SIL BUG - UNEXPECTED SYSTEM R CODE code FROM F-CODE = code	Internal error. Consult a systems analyst.	ALL
F 0205	SIL BUG - UNEXPECTED SYSTEM SS CODE code FROM F-CODE = code	Internal error. Consult a systems analyst.	ALL
F 0206	SIL BUG - UNEXPECTED SYSTEM CERR code FROM F-CODE = code	Internal error. Consult a systems analyst.	ALL
F 0207	SIL BUG - UNEXPECTED SYSTEM SERR code FROM F-CODE = code	Internal error. Consult a systems analyst.	ALL
F 0210	NO MATCH FOR SYSERR IN R_VCODE TABLE	The SYSERR code passed to the internal routine Q5_PERR is not in the T_RCODES table. Consult a systems analyst.	
F 0211	NO MATCH FOR VCODE IN T_MVCT	A VCODE found in the T_RCODES table is not in the T_MVCT table within internal routine Q5_PERR. Consult a systems analyst.	ALL
F 0250	REQUIRED PARAMETER parameter MISSING	The user omitted a required parameter. Refer to the call description.	ALL
F 0251	DUPLICATE FILE NAME name	The user specified a file name which already has a FIT.	SIL
F 0252	REQUIRED PARAMETER FOR SET n MISSING	The user did not specify a required parameter identifying the file or the action to be performed by the call.	ALL
F 0253	INVALID LFN lfn	SIL did not recognize the specified file name.	ALL
F 0254	INVALID FLUN rflun	A FIT does not currently exist having the specified file logical unit number. To obtain a file's flun, specify the RFLUN= parameter on the Q5DEFINE, Q5GENFIT, or Q5RREQUEST call that generates the file's FIT.	SIL
F 0255	MUTUALLY EXCLUSIVE PARAMETERS	The user specified two or more mutually exclusive parameters. Refer to the call description to determine which parameter to omit.	ALL

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 0261	TOO MANY PARAMETERS	The user specified more than 199 parameters on the call.	ALL
F 0262	ILLEGAL MNEMONIC FOR PARAMETER	The user specified an invalid value for the parameter specified by keyword.	Q5DCDPFI Q5LFIPOL Q5LFIPRI Q5LFIPUB
W 0300	MORE FILES TO LIST	More file indices exist than can fit in the buffer area SIL defines. Consult a systems analyst.	Q5GETPFI Q5LFIPOL Q5LFIPRI Q5LFIPUB
F 0301	ILLEGAL POOL	The caller is not a member of the specified pool, or the pool is not attached.	Q5LFIPOL
W 0304	NO FILES QUALIFY	No files match the qualifiers specified on the call.	Q5LFIPOL Q5LFIPRI Q5LFIPUB
F 0310	NON-PRIVILEGED USER	A nonprivileged user attempted to issue a privileged call.	Q5GETPFI
F 0311	packid DISK NOT UP	The specified disk pack is not currently available to the system.	Q5GETPFI
F 0312	packid PACKID NOT FOUND	The specified disk pack is not currently available.	Q5GETPFI
F 0320	ILLEGAL MESSAGE LENGTH	The message length is zero.	Q5SNDMCR Q5SNDMCE Q5SNDMDF Q5SNDMJC Q5SNDMOP
F 0321	ILLEGAL DESTINATION	The call specified a controller or controllee that does not exist.	Q5MSGCTR Q5SNDMCE Q5SNDMCR Q5SNDMJC Q5SNDSTR
F 0322	LOGGED OUT TERMINAL	The controller specified in the request is a logged out terminal.	Q5SNDMCR
F 0323	DIFFERENT SUFFIX	The controller specified in the request is a terminal which is now logged on under a different suffix.	Q5SNDMCR Q5SNDMJC
F 0324	SYSTEM BUFFER BUSY	The system buffer is busy. Try again later.	Q5SNDMCR Q5SNDMJC
F 0325	CONTROLLEE BUSY	The controllee which is the destination of the message already has text from a controller.	Q5SNDMCE
F 0326	NO OPERATOR COMMUNICATION	Either the operator is not logged on or the system buffer is full. If the user specified the SAVE parameter on the call, the message is stored in the save table for later access by the operator.	Q5SNDMOP
F 0327	DAYFILE FULL	The user cannot send more messages to the dayfile.	Q5SNDMDF
F 0328	DAYFILE NOT OPEN	The dayfile is not open for implicit I/O. Consult a systems analyst.	Q5SNDMDF

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 0329	ILLEGAL VBA FOR Q5DAYFLE	Illegal virtual byte address for dayfile.	Q5SNDMDF
F 0330	DAYFILE NOT FOUND	The system cannot find the Q5DAYFLE file. Q5SNDMDF cannot be called from an interactive job. If the call was from a batch job, consult a systems analyst.	Q5SNDMDF
F 0337	ERROR IN SENDING MESSAGE TO DAYFILE	The batch processor received a message that it could not put in the dayfile.	Q5SNDMCR Q5SNDMJC
F 0340	ILLEGAL BUFFER LENGTH	The user's message buffer is too short or too long.	Q5GETMCR Q5GETMOP Q5GETMCE
F 0341	NO MESSAGE AVAILABLE	No message is waiting for this task.	Q5GETMCR Q5GETMOP Q5GETMCE
F 0342	TOO MANY DELIMITERS	SIL encountered more than 200 delimiters in the message.	Q5GETMCE Q5GETMCR Q5GETMOP
F 0343	LEVEL 1 TASK	The caller is a level 1 task and therefore cannot have a controller from which to obtain a message.	Q5GETMCR
F 0344	CONTROLLER MESSAGE WAITING	A message cannot be transmitted because the task has a message from a controller waiting.	Q5GETMCE
F 0345	CONTROLLEE WAITING FOR MESSAGE	The controllee from which a message is expected is waiting.	Q5GETMCE
F 0350	CONTROLLEE ALREADY EXISTS	The controllee cannot be initiated because a controllee already exists.	Q5INIT Q5INITCH
F 0351	CONTROLLEE filename NOT FOUND	The controllee file filename does not exist.	Q5INIT Q5INITCH
F 0352	NOT ENOUGH TIME	There is insufficient time to run the controllee.	Q5INIT Q5INITCH
F 0353	ILLEGAL PRIORITY	An illegal priority value was specified.	Q5INIT Q5INITCH
F 0354	DROPPFILE CREATE ERROR	An error was encountered when attempting to create the dropfile.	Q5INIT Q5INITCH
F 0355	filename NOT EXECUTABLE	The controllee program file filename is not executable.	Q5INIT Q5INITCH
F 0356	filename IO ERROR	A mass storage error was encountered when attempting to read the controllee program file filename.	Q5INIT Q5INITCH
F 0357	SYSTEM TABLES FULL	Controllee cannot be initiated because the system tables are full. Try again later.	Q5INIT Q5INITCH
F 0358	filename ABNORMALITY	The controllee file filename cannot be initiated because of an abnormality in the file or in the dropfile I/O number.	Q5INIT Q5INITCH

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 0359	TOO MANY LEVELS	Controllee cannot be initiated because the controllee chain already contains nine tasks. An interactive user can initiate a chain of eight tasks. A batch job can initiate a chain of only seven tasks because it is the second task in the chain.	Q5INIT Q5INITCH
F 0360	DROPPFILE TOO SMALL	Controllee cannot be initiated because the drop file is too small.	Q5INIT
F 0361	PERSISTENT DROPPFILE	System unable to destroy existing dropfile.	Q5INIT
F 0362	INTERRUPT TABLE FULL	Controllee cannot be restarted because the interrupt Register Table is full. Try again later.	Q5INIT
F 0363	DROPPFILE VERIFY ERROR	Controllee cannot be initiated because the drop file cannot be verified.	Q5INIT
F 0364	DISK READ BUFFER FULL	Insufficient system buffer space to initiate task. Try again later.	Q5INIT
F 0365	filename BAD MINUS PAGE	The controllee file (filename) contains a bad minus page.	Q5INIT
F 0366	SYSTEM BUG-DROPPFILE VERIFICATION	System detected an undefined error in drop file verification. System error.	Q5INIT
F 0367	PRIVILEGED OPEN	Controllee program file is currently open (using a privileged OPEN) to a privileged user.	Q5INIT Q5INITCH
F 0370	NO CONTROLLEE TO DISCONNECT	No controllee exists to disconnect.	Q5TERMCE
F 0375	EXCESSIVE CALLS	The task attempted to call the Q5CPUTIM routine more times than allowed by the installation parameter setting.	Q5CPUTIM
F 0380	INTERRUPT ADDRESS ERROR	The program interrupt address is greater than the virtual address range.	Q5DISAMI Q5ENAMI
F 0381	INTERRUPT OR DATA BASE ADDRESS ERROR	The program interrupt address or the data base address is greater than the virtual address range.	Q5ENATI Q5RFI
F 0382	DATA BASE LENGTH OUT OF RANGE	The user specified a data base length that is out of range.	Q5ENATI
F 0383	USER NOT IN INTERRUPT MODE	The user attempted to get interrupted program information when the program had not been interrupted.	Q5GETIIP Q5GETIRF
W 0391	REQUESTED LARGE PAGE LIMIT EXCEEDS MAX LARGE PAGE LIMIT	The current large page limit specified on the Q5SETLP call exceeded the maximum large page limit for the task. Therefore, the current large page limit was set to the maximum large page limit.	Q5SETLP
F 0400	lfn DOES NOT EXIST	The specified batch file lfn does not exist.	Q5DESBIF Q5RUNBIF
F 0410	ILLEGAL MESSAGE	The program cannot issue the Recall system message.	Q5RECALL
F 0420	NON INTERRUPT ROUTINE	The program issuing the Q5RFI call is not an interrupt subroutine.	Q5RFI

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 0450	ADDRESS OUT OF USER VIRTUAL RANGE	The caller specified an address outside the user's virtual address range with the OUTADDR=, OUTDESC=, INADDR=, or INDESC= parameter.	Q5ADVISE
F 0451	ILLEGAL LENGTH	The length specified by the user was either too long or zero.	Q5ADVISE
W 0452	ADVISE IN	Only part of the virtual space requested was paged in because of insufficient memory to accomodate the entire specified virtual range.	Q5ADVISE
W 0453	PARTIAL ADVISE OUT	Only part of the virtual space specified as no longer needed was paged out because a page within the specified range was locked down.	Q5ADVISE
W 0454	PARTIAL ADVISE - ERRORS 0452 AND 0453	Only part of the virtual space replacement was performed. The reasons are given under errors 0452 and 0453.	Q5ADVISE
W 0455	PAGE ALREADY IN CORE	A page of the virtual space requested paged in was already in memory. The rest of the space was paged in.	Q5ADVISE
W 0456	PARTIAL ADVISE - ERRORS 0452 AND 0455	Only part of the request space was paged due to errors 0452 and 0455.	Q5ADVISE
W 0457	PARTIAL ADVISE - ERRORS 0453 AND 0455	Only part of the requested space was paged due to errors 0453 and 0455.	Q5ADVISE
W 0458	PARTIAL ADVISE - ERRORS 0452, 0453, AND 0455	Only part of the request space was paged due to errors 0452, 0453, and 0455.	Q5ADVISE
F 0459	ADVISE OUT ADDRESS MISSING	The length of the space to be paged out was specified, but not its address.	Q5ADVISE
F 0460	ADVISE IN ADDRESS MISSING	The length of the space to be paged in was specified, but not its address.	Q5ADVISE
F 0461	ADVISE OUT LENGTH NOT POSITIVE	A positive value must be specified as the length of the area to be paged out.	Q5ADVISE
F 0462	ADVISE IN LENGTH NOT POSITIVE	A positive value must be specified as the length of the area to be paged in.	Q5ADVISE
F 0463	ADVISE OUT PAGE COUNT TOO LARGE	The length of the space to be paged out is too large to fit in the appropriate field of the system message.	Q5ADVISE
F 0464	ADVISE IN PAGE COUNT TOO LARGE	The length of the space to be paged in is too large to fit in the appropriate field of the system message.	Q5ADVISE
F 0470	CALL NOT VALID FROM THIS TASK	The task cannot issue a Q5VRACC call because it is not public or its variable rate permit flag is not set.	Q5VRACC
F 0471	VARIABLE RATE ACCOUNTING NOT VALID AT THIS INSTALLATION	The site has set an installation parameter preventing use of variable rate accounting.	Q5VRACC
F 0472	CALL NOT VALID ON THIS SYSTEM	The site did not install variable rate accounting on the system.	Q5VRACC

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 0480	NO CONTROLLEE TO LIST	No controllee was found.	Q5GETCEN
F 0485	NO CONTROLLER TO LIST	No controller was found. This message can be received by an interactive job, but not by a batch job.	Q5GETCRN
F 0504	ILLEGAL PFI ORDINAL	The ENTRY= parameter is either zero or greater than the number of entries in the SIL-defined area.	Q5DCDPFI
F 0505	ILLEGAL ROUTINE USAGE	The user did not call the Q5GETPFI routine before issuing this call and did not specify the MYFILE= parameter on this call.	Q5DCDPFI Q5DCDPLB
F 0506	STLEN WRONG SIZE	An incorrect length was specified for the file segment table.	Q5DCDPFI
F 0508	MYFILE WRONG SIZE	An incorrect length was specified by the MYLEN= parameter for the file index entry length.	Q5DCDPFI
F 1400	SIX I/O REQUESTS STILL PENDING FOR FILE lfn	The user cannot issue another I/O request for the specified file until one of its outstanding file requests completes. The user should issue a Q5CHECK call specifying that SIL wait until the request completes before returning control to the caller.	Q5READ Q5REWIND Q5SKIP Q5WRITE
F 1401	NO I/O BUFFER SPECIFIED FOR FILE lfn	The user did not specify an I/O buffer in the file's FIT or on the I/O request.	Q5READ Q5WRITE
F 1402	FILE lfn DOES NOT EXIST	SIL cannot purge or return the file because it does not exist.	Q5PURGE Q5RETURN
W 1403	BAD "RSN" SPECIFIED	The user specified a request serial number that does not identify a pending I/O request. To obtain the number assigned to the request, specify the RSN= parameter on the Q5READ or Q5WRITE call.	Q5CHECK
W 1404	UNCLEARED ERROR ON PREVIOUS TAPE	The user did not check for error status for the previous I/O operation on the specified file. If the user did not specify the WAIT parameter on the Q5READ or Q5WRITE call, he must issue a Q5CHECK call to check for error status. If the user specifies the WAIT parameter, he must also specify the STATUS= parameter to check the error status.	Q5READ Q5WRITE
F 1405	FILE lfn NOT CURRENTLY OPEN	The user has not opened the specified file for I/O. Issue a Q5OPEN call specifying the file.	Q5GETP Q5READ Q5REWIND Q5SKIP Q5WRITE
F 1407	I/O STILL PENDING FOR FILE lfn	The user cannot close the file because SIL has not completed implicit I/O for the file. If the user is nonprivileged, the message could indicate a system error. Consult a systems analyst.	Q5CLOSE
W 1408	FILE lfn IS OPEN TO ANOTHER PROGRAM OF THIS USER	Because the specified file is open to another program executing under this user number, SIL can close the file for this program, but cannot destroy or give the file.	Q5CHECK Q5READ Q5WRITE

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 1409	BUFFER SIZE <1 OR >24 BLOCKS FOR FILE lfn	The I/O buffer must be between 1 and 24 512-word blocks long. Correct the specified buffer size on the call.	Q5CHECK Q5READ Q5WRITE
F 1410	DATA QUANTITY EXCEEDS BUFFER SIZE	SIL read a tape record longer than the I/O buffer for the specified file.	Q5CHECK Q5READ
F 1411	LARGE-PAGE BUFFER FOR FILE lfn IS GREATER THAN 128 BLOCKS	When the user specifies that SIL is to use a large page buffer, it must be 128 blocks long. Correct the buffer size.	Q5CHECK Q5OPEN Q5READ Q5WRITE
F 1412	ILLEGAL SKIP ON U-TYPE FILE lfn	SIL cannot skip logical partitions (records, groups, or files) on a U-format file. It can skip physical blocks.	Q5SKIP
F 1413	SKIP FORWARD ILLEGAL. LAST OP. WAS WRITE ON FILE lfn	The user cannot issue a Q5SKIP call to skip forward on a file when the last operation on the file was a write operation. The user can request a skip backward.	Q5SKIP
W 1414	BEGINNING OF INFORMATION ENCOUNTERED, FILE lfn	SIL cannot skip further backward as it has reached the beginning of information for the file.	Q5SKIP
F 1415	SKIP OF GROUPS ILLEGAL FOR F-TYPE FILE, NAME=lfm	The F file format does not allow skipping by groups because group delimiters do not exist in F format.	Q5SKIP
F 1416	EOF ENCOUNTERED ON FILE lfn	SIL has read the end of file indicator.	Q5READ Q5SKIP Q5WRITE
W 1417	STATUS OF LAST I/O NOT CHECKED FOR FILE lfn	The user did not check for error status after the last I/O operation. If the user did not specify the WAIT parameter on the Q5READ or Q5WRITE call, he must issue a Q5CHECK call to check for error status. If the user specifies the WAIT parameter, he must also specify the STATUS= parameter to check the error status.	Q5REWIND
W 1418	GROUP OR FILE DELIMITER UNDEFINED FOR FILE lfn	The user cannot write group or file delimiters or skip by groups or files on the specified file because its format does not delimit groups or files.	Q5ENDPAR Q5SKIP
W 1419	BEGINNING OF FILE ENCOUNTERED FOR FILE lfn	SIL cannot skip further backward because it is positioned at the beginning of the file.	Q5SKIP
F 1420	SIO ERROR - BLK. NO IS NEG. BEFORE SKIP FOR FILE lfn	Internal SIL error. Consult a systems analyst.	Q5SKIP
F 1421	FILE lfn NOT IMPLICITLY OPEN	The user cannot issue a Q5MAPIN or Q5MAPOUT call for the specified file because it has not been opened for implicit I/O.	Q5MAPIN Q5MAPOUT
F 1422	FILE lfn FUNCTION ABORTED BY THE STATION OPERATOR	The station operator aborted the function being performed on the specified file. If needed, request an explanation from the station operator.	Q5READ Q5SKIP Q5STATUS Q5WRITE
F 1423	VIRTUAL FILE lfn CANNOT BE LESS THAN 2 PAGES	The user cannot create a virtual code file shorter than two 512-word blocks or reduce an existing block to less than two 512-word blocks.	Q5DEFINE Q5REDUCE Q5REQUEST

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 1424	FILE lfn IS NOT ATTACHED	SIL cannot perform the requested operation on an unattached permanent file. The user must attach the file.	Q5CHANGE Q5GIVE Q5PURGE Q5REDUCE Q5RETFIT Q5ROUTE
F 1426	Q5RQUEST REQUIRED BEFORE TAPE FILE lfn CAN BE OPENED	The user must call Q5RQUEST to assign the tape file to the job before issuing the Q5OPEN call to open the tape file for I/O.	Q5OPEN
W 1427	FILE lfn IS ALREADY OPENED.	The specified file was open before this Q5OPEN call.	Q5OPEN
F 1429	lfn MUST BE A VIRT. CODE FILE TO CHANGE THE DROPFILE LEN.	SIL cannot change the drop file length when the file is a physical data file. Remove the DFLEN= parameter from the Q5CHANGE call.	Q5CHANGE
F 1431	WRITE ATTEMPTED ON WRITE-PROTECTED TAPE FILE lfn	The user issued a Q5WRITE call for a tape file without a write ring inserted in the tape volume. Request the operator insert a write ring in the volume.	Q5CHECK Q5WRITE
F 1432	NO "WSA=" DEFINED FOR FILE lfn	The user cannot perform explicit I/O by logical partitions without specifying a working storage area for the file.	Q5GETN Q5GETP Q5PUTN
F 1433	READ NOT ALLOWED ON FILE lfn; NO READ ACCESS	SIL cannot read the specified file because the user did not open the file for read access.	Q5READ
F 1434	END OF INFORMATION ENCOUNTERED ON FILE lfn	SIL encountered the end of information for the file. The user must specify the STATUS= parameter on the call to check for the end of information.	Q5GETN Q5GETP Q5PUTN Q5PUTP
W 1435	RECORD LENGTH OUTSIDE MIN/MAX RANGE FOR FILE lfn	SIL transferred a record shorter or longer than the range of record lengths specified by the mnr and mxr fields in the FIT.	Q5GETN Q5PUTN Q5PUTP
W 1436	DATA QUANTITY EXCEEDS WSL FOR FILE lfn	The length of the partition requested exceeds the working storage area length. SIL truncated the partition, discarding the excess data.	Q5GETN
F 1437	GET FOLLOWS OUTPUT OPERATION ON FILE lfn	The user cannot issue a Q5GETN or Q5GETP call for the specified file because the last operation on the file was an output operation.	Q5GETN Q5GETP
F 1438	CONTROL WORD PARITY ERROR ON FILE lfn	SIL read a control word with even, rather than odd parity. This could indicate that the file is not a W format file.	Q5GETN Q5GETP
F 1439	CONTROL WORD FIELD ERROR ON FILE lfn	SIL read a control word with a field error. Either the control word was written incorrectly or the file does not contain W format records.	Q5GETN Q5GETP Q5PUTN
W 1440	END OF RECORD ENCOUNTERED ON FILE lfn	While reading partial records, SIL read the end of the current record.	Q5GETP
W 1441	END OF GROUP ENCOUNTER ON FILE lfn	While reading records, SIL read the end of group delimiter.	Q5GETN Q5GETP
F 1442	WRITE NOT ALLOWED ON FILE lfn; NO WRITE ACCESS	SIL cannot write on the specified file because it was not opened for write access.	Q5WRITE

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 1443	PUT NOT ALLOWED AFTER EOF ON FILE lfn; F/U TYPE RECORDS	SIL cannot write data after the end of file in an F or U format file because the F and U formats do not have file delimiters.	Q5PUTN Q5PUTP
F 1444	PART=GROUP FOR RT=U or F ILLEGAL; FILE lfn	SIL cannot read, write or skip groups on the specified F or U format file because the F and U formats do not have group delimiters.	Q5ENDPAR Q5GETN Q5GETP Q5PUTN Q5PUTP Q5SKIP
F 1446	WRITE ACCESS DENIED FOR THIS BUFFER FOR FILE lfn	The user mapped in the buffer area for read access only and then issued a Q5READ call to write data into the buffer.	Q5READ
F 1447	READ ACCESS DENIED FOR THIS BUFFER FOR FILE lfn	The user mapped in the buffer area for write access only and then issued a Q5WRITE call to read data from the buffer.	Q5WRITE
W 1448	MAX LENGTH OF FILE lfn LESS THAN REQUESTED LENGTH	The user specified a file length on the Q5GETFIL call longer than the maximum allowed file length. Specify a smaller file length.	Q5GETFIL
F 1500	INVALID ACCOUNT NUMBER	The account number specified on the Q5GIVE call is not valid.	Q5GIVE
F 1501	FILE lfn IS ATTACHED TO ANOTHER SUFFIX	SIL cannot attach the specified file because it is currently attached to another job executing under this user number.	Q5ATTACH
F 1502	FILE lfn ALREADY EXISTS AS A LOCAL FILE	SIL cannot attach the specified permanent file because a local file with that name is currently assigned to the job.	Q5ATTACH
W 1503	POOL pname ALREADY ATTACHED BY THIS USER	Either SIL cannot attach the specified pool because it is already attached to the user number or it cannot remove pool access privileges from a user because the pool is attached.	Q5PATACH Q5PREACC
W 1504	FILE lfn ALREADY ATTACHED AS	SIL cannot attach the specified permanent file because it is already attached to the job.	Q5ATTACH
F 1505	FILE lfn ALREADY EXISTS	SIL cannot create a file with or change a file's name to the specified file name because a file with that name already exists.	Q5CHANGE Q5DEFINE Q5REQUEST
W 1506	POOL NAME pname IS NOT ATTACHED	SIL cannot detach the specified pool because it is attached to the user number.	Q5PDTACH
F 1507	DUPLICATE POOL NAME pname	SIL cannot add the specified pool name to the pool list because it already exists in the pool list.	Q5CREAT
F 1508	INVALID POOLNAME pname	The pool name does not conform to the pool naming conventions (one through eight characters, beginning with a letter).	Q5PATACH Q5PCREAT Q5PDESTR Q5PGRACC Q5PREACC Q5USERL
F 1509	INVALID USER NUMBER	The user specified the input queue manager user number on the Q5GIVE call. To give a file to the input queue manager, the user must specify the IQM and ACCT= parameters.	Q5GIVE

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 1510	UNABLE TO ATTACH ALL FILES	SIL could not attach one or more of the files belonging to the user number.	Q5ATTACH
F 1511	CAN'T CREATE POOL pname - SYSTEM POOL LIST IS FULL	SIL could not add the specified pool name to the list because there is no space in the list for another name. Consult a systems analyst or destroy a pool for which you are pool boss.	Q5PCREAT
F 1512	PART=T ILLEGAL FOR LABELED TAPE FILE lfn.	The user requested the writing of a tape mark on a labeled tape. Q5ENDPAR can only write tape marks on unlabeled tapes.	Q5ENDPAR
F 1513	FILE NAME NOT SPECIFIED FOR MULTIFILE SET setname	The user must specify the file name, rather than a flun number, when opening a file in a multifile set.	Q5OPEN
F 1516	VIRTUAL ADDRESS OVERLAP DURING MAP-IN/OUT OF DROPFILE	Either the user specified an area of virtual space to be mapped in to the dropfile that is already mapped in to another file or specified an area to be mapped out that is not mapped in to the file.	Q5MAPIN Q5MAPOUT
F 1517	ILLEGAL USER NUMBER FOR USER1 FUNCTION ON FILE lfn	The user number does not have USER1 privileges, but was attempting a USER1 function on the specified file.	Q5ATTACH Q5OPEN
F 1518	CANNOT PRIVILEGE OPEN ATTACHED FILE lfn	A nonprivileged user attempted a privileged open.	Q5OPEN
F 1519	VIRTUAL ADDRESS OVERLAP ON FILE lfn	Either the user specified an area of virtual space to be mapped in to the specified file that is already mapped in to another file or specified an area to be mapped out that is not mapped in to the drop file.	Q5MAPIN Q5MAPOUT
F 1520	FILE lfn OPENED IN READ-ONLY MODE	SIL cannot perform the requested function because the file was not opened for write access.	Q5ENDPAR Q5PUTN
F 1522	UNABLE TO PROCESS FILE lfn, TOO MANY ACTIVE FILES	SIL cannot create or open the specified file because 70 files (the operation system limit) are already active for this task.	Q5DEFINE Q5OPEN Q5REQUEST
F 1525	SECURITY LEVEL OF FILE lfn TOO HIGH	One of the following. - The user cannot open the specified file because its security level is higher than the maximum security level the user number is assigned. - The user cannot give the specified file to the specified user because its security level is higher than the maximum security level the other user is assigned.	Q5GIVE Q5OPEN
F 1526	USER DIRECTORY OR POOL WAS NOT FOUND FOR FILE lfn	This message probably indicates a system error. Consult a systems analyst.	Q5OPEN
F 1527	FUNCTION NOT ALLOWED; FILE lfn IN USE BY PRIVILEGED USER	The user cannot attach or open the specified file because a privileged user has opened the file without specifying shared access.	Q5ATTACH Q5OPEN
F 1528	FILE lfn NOT OPENED; NO MORE WRITE OPENS PERMITTED	The user cannot open another file for write access without closing one of the files currently open for write access.	Q5OPEN

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 1529	FILE lfn NOT OPENED; NO MORE ROOM IN USER TABLE	The system cannot enter the specified file in the user table. Consult a systems analyst.	Q5OPEN
F 1531	UNABLE TO DESTROY POOL pname	The user cannot destroy the specified pool because he or she is not the pool boss or because he or she or another user has the pool attached.	Q5PDESTR
F 1537	NOT ENOUGH ROOM IN DROPFILE	The user can map no more virtual space into the drop file.	Q5MAPIN
F 1539	BLOCK COUNT OF FILE lfn NOT A MULTIPLE OF 128	SIL cannot perform explicit I/O in large page units unless the virtual region length is a multiple of 128 blocks.	Q5MAPIN
F 1541	FILE INDEX COPY FOR FILE lfn IS OUT OF BOUNDS	The user specified an array containing the File Index entry that is outside the virtual address space the user is permitted to access.	Q5OPEN
F 1544	CANNOT ATTACH pname, ALREADY ATTACHED TO 4 POOLS	The user number has four pools attached and so cannot attach another pool. To attach the specified pool, detach one of the attached pools.	Q5PATTACH
F 1545	CANNOT ATTACH POOL pname - USER HAS NO ACCESS	The pool boss for the specified pool has not granted pool access to this user number. Request pool access from the pool boss.	Q5PATTACH
F 1546	USER IS NOT THE POOL BOSS FOR POOL pname	SIL can only perform the requested function when the caller is the pool boss for the specified pool.	Q5GIVE Q5PDESTR Q5PGRACC Q5PREACC Q5PURGE
F 1548	NO READ ACCESS SPECIFIED FOR FILE lfn	The user cannot map in a file that is not opened for read access.	Q5MAPIN
F 1550	VIRTUAL ADDRESS OF FILE lfn SAME AS THAT IN ADVISE CALL	The user attempted to access a virtual address that he had previously notified the system that he would not access (via an Q5ADVISE call).	Q5MAPIN Q5MAPOUT
F 1551	MASS STORAGE ADDR+LENGTH EXCEEDS LENGTH OF FILE lfn	The virtual region length starting at the specified mass storage sector exceeds the length to which the file can extend.	Q5MAPIN Q5MAPOUT
F 1553	A VIRTUAL ADDRESS FOR FILE lfn NOT ON LARGE PAGE BOUNDARY	The user requested implicit I/O using large page units, but did not specify the LOAD utility parameter to load the virtual region on the large page boundary.	Q5MAPIN Q5MAPOUT
F 1560	FUNCTION FAILED FOR FILE lfn; BOUND IMPLICIT MAP FULL	The user cannot map in another virtual space region into the specified file without mapping out a mapped in region or combining two mapped in regions.	Q5MAPIN Q5MAPOUT Q5OPEN
F 1561	CANNOT PERFORM FUNCTION ON FILE lfn; PAGES STILL LOCKED IN	SIL cannot map out the virtual region because it is currently performing implicit I/O from that region. If the user is nonprivileged, this message could indicate a system error. Consult a systems analyst.	Q5MAPOUT
F 1562	SPACE UNDEFINED AT MAPOUT FOR FILE lfn	SIL cannot map out the virtual region because that region was not mapped in to the specified file.	Q5MAPOUT

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine																										
F 1565	INCORRECT LENGTH OF VIRTUAL REGION FOR FILE lfn	The user specified a longer virtual region to be mapped out than the region originally mapped in.	Q5MAPOUT																										
F 1566	DROP FILE MAP FULL	SIL cannot map out another area to the drop file.	Q5CLOSE Q5MAPOUT																										
F 1580	MULTIPLE STATION (SERR) ERROR CODES; VALUE = value	<p>The peripheral (station) operating system detected more than one error condition. SIL combined the codes for the error conditions using inclusive OR operations to form the value in the message. Each bit set indicates an error condition. The hexadecimal values are listed below.</p> <table><tr><th>Value</th><th>Condition</th></tr><tr><td>1</td><td>Device not ready.</td></tr><tr><td>2</td><td>Tape, station buffer unit, or transmission parity error.</td></tr><tr><td>4</td><td>Data quantity exceeds user buffer size.</td></tr><tr><td>8</td><td>End of tape encountered.</td></tr><tr><td>10</td><td>End of file encountered.</td></tr><tr><td>20</td><td>Attempted to write on a tape without a write ring.</td></tr><tr><td>40</td><td>Disk channel failed.</td></tr><tr><td>80</td><td>Tape data lost.</td></tr><tr><td>100</td><td>Attempted backspace from load point.</td></tr><tr><td>200</td><td>Mass storage positioning error.</td></tr><tr><td>400</td><td>Station operator aborted function.</td></tr><tr><td>800</td><td>File extension error.</td></tr></table>	Value	Condition	1	Device not ready.	2	Tape, station buffer unit, or transmission parity error.	4	Data quantity exceeds user buffer size.	8	End of tape encountered.	10	End of file encountered.	20	Attempted to write on a tape without a write ring.	40	Disk channel failed.	80	Tape data lost.	100	Attempted backspace from load point.	200	Mass storage positioning error.	400	Station operator aborted function.	800	File extension error.	Q5READ Q5REWIND Q5SKIP Q5STATUS Q5WRITE
Value	Condition																												
1	Device not ready.																												
2	Tape, station buffer unit, or transmission parity error.																												
4	Data quantity exceeds user buffer size.																												
8	End of tape encountered.																												
10	End of file encountered.																												
20	Attempted to write on a tape without a write ring.																												
40	Disk channel failed.																												
80	Tape data lost.																												
100	Attempted backspace from load point.																												
200	Mass storage positioning error.																												
400	Station operator aborted function.																												
800	File extension error.																												
F 1587	END OF TAPE REACHED FOR FILE lfn	SIL read the end of tape indicator for the specified file. If the file is a multivolume set, SIL read the end of tape indicator on the last volume.	Q5CHECK Q5OPEN Q5READ Q5SKIP																										
F 1588	TAPE, STATION BUFFER UNIT OR PARITY ERROR FOR FILE lfn	SIL encountered one of these errors for the specified file.	Q5CHECK Q5READ Q5REWIND Q5SKIP Q5WRITE																										
F 1589	DEVICE NOT READY FOR FILE lfn	The device on which the file resides is not ready to transfer data. Request that the operator ready the device.	Q5CHECK Q5READ Q5REWIND Q5SKIP Q5WRITE																										

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
W 1593	ATTEMPTED TO BACKSPACE FILE lfn AT LOAD POINT	SIL cannot backspace the specified file because it is positioned at load point.	Q5REWIND Q5SKIP
F 1596	ERROR IN POSITIONING MASS STORAGE DEVICE FOR FILE lfn	Hardware error. Consult a systems analyst.	Q5CHECK Q5READ Q5REWIND Q5SKIP Q5WRITE
F 1598	CANNOT PERFORM FUNCTION ON FILE lfn - I/O CHANNEL FAILED	SIL could not perform the requested function because the I/O channel failed. Consult a systems analyst.	Q5CHECK Q5READ Q5WRITE
F 1605	LARGE PAGE BUFFER FOR FILE lfn NOT ON LARGE PAGE BOUNDARY	The user specified a 128-block (large page) I/O buffer for the file, but did not specify the LOAD utility parameter required to load the buffer on a large page boundary.	Q5OPEN
F 1620	ATTEMPT TO IMPLICITLY OPEN FILE lfn WITH WRITE ONLY ACCESS	SIL cannot open a file for implicit I/O to which the user does not have read access.	Q5OPEN Q5READ Q5WRITE
W 1621	USER TAPE VOLUME SWITCHED FOR FILE lfn	SIL read the end of tape indicator and switched to the next volume in the specified file set.	Q5OPEN Q5READ Q5WRITE
W 1622	NEW TAPE VOLUME ASSIGNED BY SYSTEM FOR FILE lfn	The operator assigned an available tape volume to the specified file set after SIL read the end of tape indicator. The user specified a set identifier but did not specify any volume serial numbers.	Q5OPEN
F 1623	LABEL PARAM.PASSED FOR UNLABELED TAPE FILE lfn	The user specified a label field parameter when opening an unlabeled tape file.	Q5OPEN
F 1644	POOLNAME pname IS NOT ATTACHED OR CALLER NOT POOL MEMBER	The user cannot give a file to the pool either because the pool is not attached or because the pool boss has not granted the user access to the pool.	Q5GIVE
F 1650	BUFFER NOT ON 512-WORD PAGE BOUNDARY FOR FILE lfn	The I/O buffer for the specified file is not on a page boundary. The user must specify the LOAD utility parameter to load the buffer on a page boundary.	Q5OPEN
F 1653	BUFFER FOR FILE lfn IN UNASSIGNED VIRTUAL SPACE	SIL error. Consult systems analyst.	Q5CHECK Q5READ Q5WRITE
F 1658	MAPS= + LEN= FOR FILE lfn IS BEYOND MAX. VIRTUAL SPACE	The specified virtual region extends beyond the maximum virtual address the user can access.	Q5OPEN
F 1679	NEW FILE LENGTH FOR FILE lfn GREATER THAN EXISTING LENGTH	The user specified the reduced file length to be greater than the existing file length.	Q5REDUCE
F 1680	FILE lfn ALREADY EXISTS AT DESTINATION	The user cannot give the specified file as requested because the destined owner (another user, a pool, or the public file list) already has a file with that name.	Q5GIVE
F 1681	FILE lfn SAME NAME AS PUBLIC	The user cannot give the specified file because it has the same name as a public file.	Q5GIVE
F 1682	UNDEFINED USER NUMBER usernum	The user specified a nonexistent user number on the Q5GIVE call.	Q5GIVE

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 1683	OUTPUT FILE lfn IMPROPERLY NAMED	Print file names must be in the format Pnnxxxxx where nn is two digits indicating the position of the file within a family of print files and xxxxx is the family name.	Q5GIVE
F 1684	USER IS NOT A USER-1 ROUTINE	Only USER-1 routines can specify the SADDR= parameter on a Q5GIVE call.	Q5GIVE
F 1685	FUNCTION FAILED OF FILE lfn USER NOT PRIVILEGED	SIL cannot perform a privileged function requested by a nonprivileged user number.	Q5CLOSE Q5DEFINE Q5GIVE Q5OPEN Q5PURGE Q5REQUEST
F 1686	FILE lfn IS A CONTROLLEE OR DROPFILE	SIL cannot transfer ownership of a controllee or drop file.	Q5GIVE
F 1687	DISK IS LOGICALLY OFF FOR FILE lfn	The disk on which the file resides is not currently available to the system. Ask the operator to logically turn on the disk.	Q5DEFINE Q5GIVE Q5OPEN Q5PURGE Q5RETURN Q5REQUEST
F 1688	FUNCTION ON FILE lfn WOULD EXCEED FILE SPACE LIMIT	SIL did not create a local file or give a file as requested by the user because the file space of the user would be exceeded.	Q5DEFINE Q5GIVE Q5REQUEST
F 1689	VRI= NOT VALID FOR NON-CODE FILE	The user cannot specify a variable rate index for a data file.	Q5GIVE Q5REQUEST
F 1690	FILE lfn STILL OPEN TO ANOTHER TASK	SIL cannot purge or return a file that is open to another active program or is privileged opened.	Q5PURGE Q5RETURN
F 1700	NO DISPOSITION CODE FOR FILE lfn	To route a file, the user must specify a disposition code for the file using the DC= parameter.	Q5ROUTE
F 1701	ILLEGAL DISPOSITION CODE xx	SIL does not recognize the specified disposition code.	Q5ROUTE
F 1702	ILLEGAL SITE IDENTIFIER site, OR SITE NOT LOGGED IN	SIL either does not recognize the site identifier or the specified site is not logged in to the system.	Q5ROUTE
F 1703	FILE lfn TOO BIG FOR SERVICE STATION	SIL cannot route the specified file because it is too large for transfer by the service station. The user must split the file into two or more files and route each file.	Q5ROUTE
F 1711	NO MASS STORAGE SPACE FOR FILE	The system has no mass storage space available for creating the specified file.	Q5DEFINE Q5REQUEST
F 1712	OPERATOR-INITIATED ERROR FOR FILE lfn	The operator entered a command preventing creation of the specified file.	Q5DEFINE Q5REQUEST
F 1713	STANDBY JOB CANNOT REQUEST A TAPE FOR FILE lfn	Jobs assigned to the standby job class cannot request tape files. Resubmit the job with a different job class.	Q5DEFINE Q5REQUEST
F 1716	CANNOT FIND DISK PACK pack FOR FILE lfn	SIL cannot find an on-line disk pack whose name matches the pack name specified on the Q5DEFINE or Q5REQUEST call.	Q5DEFINE Q5REQUEST

TABLE B-2. SYSTEM INTERFACE LANGUAGE ERROR MESSAGES (Contd)

Error Code	Message	Significance	Issuing Routine
F 1717	CANNOT SEND TAPE REQUEST MESSAGE TO THE OPERATOR	The operator is not logged in to the system. Resubmit the job when the operator is logged in.	Q5DEFINE Q5RQUEST
F 1718	ATTEMPT TO EXCEED MAXIMUM ALLOWABLE FILE SIZE FOR FILE lfn	The user specified a file length on the Q5DEFINE or Q5RQUEST call that exceeds the maximum length allowed by the installation.	Q5DEFINE Q5RQUEST
F 1720	CANNOT LOCATE THE USER OR POOL FOR FILE lfn	A privileged user specified an array on the Q5DEFINE or Q5RQUEST call containing a File Index entry, but the entry contains an invalid user number or pool name.	Q5DEFINE Q5RQUEST
F 1722	WITH FILE lfn NUMBER OF FILES EXCEEDS LIMIT FOR USER	SIL cannot create or give the specified file because the number of files belonging to the user number would exceed the limit set for the user number.	Q5DEFINE Q5GIVE Q5RQUEST
F 1724	ILLEGAL OPERATION ON TAPE FILE lfn	The user specified a tape file assigned to the job on a Q5DEFINE, Q5GIVE, Q5REDUCE, or Q5ROUTE call.	Q5DEFINE Q5GIVE Q5REDUCE Q5ROUTE
F 1725	FIT ALREADY EXISTS FOR FILE lfn	The specified file name is already associated a FIT so the user cannot generate another FIT for that file.	Q5GENFIT

- Abnormal Termination -**
The procedures the system follows when a task encounters a fatal error.
- Absolute Binary Card -**
See 80-Column Binary Card.
- Access -**
Permitted mode of use. A user could be permitted to access a file in read, write, read and write, or write temporary mode. A user could also be permitted to access a pool of files.
- Access Station -**
A CYBER computer system controlled by the NOS operating system that is used to enter jobs into the CYBER 200 system and to control peripheral devices.
- Account Identifier -**
One through eight characters indicating who is to be charged for system resources used by a job.
- Batch Deck -**
A card deck that begins with a STORE card and that ends with a card having the digits 6, 7, 8, and 9 multipunched in column 1.
- Batch Input File -**
A mass storage file containing the control statements, programs, data, and directives that define a batch job.
- Batch Job -**
A sequence of tasks executed under control of the batch processor. A batch job is initiated when the system reads a batch input file.
- BATCHPRO -**
Refer to Batch Processor.
- Batch Processor -**
A system utility that processes batch jobs. Control statements in the job having file names are executed as a controllee of the batch processor.
- Beginning of information (BOI) -**
The point in a file before which no data exists.
- Block -**
The smallest quantity of data that can be read or written by one device access. On CYBER 200 mass storage, a block is 512 64-bit words. On magnetic tape, a block is a physical record.
- Byte -**
A sequence of 8 bits that is a subdivision of a word and is sufficient to represent a single character.
- Card Reader ID Card -**
Another name for STORE card.
- Checkpoint -**
A system feature that captures a task and any of its controllees at some point into execution such that the task can be restarted from that point. Checkpoint is called through a FORTRAN program by the name CHKPNP.
- Controllee -**
A task called into execution by a controller.
- Controllee Chain -**
An ordered set of tasks. Except for the first and last tasks in the chain, each task was initiated by the task at the next higher level (its controller) and initiates the task at the next lower level (its controllee). The chain can comprise up to nine tasks, beginning with the job's controller.
- Controllee File -**
See Virtual Code File.
- Controller -**
A relative term that indicates a member of a controllee chain has a controllee task attached. A controller might be a controllee of another task.
- CPU -**
Central processing unit, the computational facility of the CYBER 200 system.
- Dayfile -**
A file produced by the batch processor for a batch job that gives a history of the job. Information on the file includes the times control statements began execution, messages sent to the dayfile by the operator or a task, and error or status information produced by system utilities. The dayfile is printed as part of job output.
- Directive -**
Supplementary control information in a file required in addition to a utility call. Directives are required, for example, with Update.
- Drop File -**
A file created by the system to contain modified pages of an executing task, free space, and write-temporary files.
- Drop file names are formed by the system shifting the controllee file name right two characters and prefixing it with digits that identify the suffix (digits 1 through 4 corresponding to suffixes A through D) and the level 1 through 9 in a controllee chain.
- End of file (EOF) -**
Within R format, an ASCII FS (#1C) character marks an end of file if the record mark is ASCII US or RS. Within W format, an end of file is a control word with the end of file bit set.

End of information (EOI) -

The point in a file after which no data exists.

Explicit Input/Output -

A means of accessing a mass storage or tape file in which data is buffered under program control.

Family -

A set of files with names that begin with Pnn which is printed at job termination when a file with a name beginning with PXX is added to the family.

File -

A collection of data that can be accessed by file name. In the absence of an adjective such as card or tape, all references to files in this manual imply mass storage files.

File Index Table (FILEI) -

A system table that holds all information relating to file characteristics. Output from the AUDIT or FILES utilities shows much of the table information.

File Type -

A category that defines file structure from a system standpoint. File types are physical data and virtual code.

Group -

A set of data within a file consisting of one or more records. Groups can exist within R or W format files. In R format, an ASCII GS (#1D) character terminates a group (if the record mark is ASCII US or *RS). In W format, an end-of-group control word terminates a group.

Implicit Input/Output -

A means of accessing a mass storage file in which the system brings a page of the file into central memory in response to a reference on that page.

Input/Output Connector (IOC) -

An entry in a minus page that links a file with a task.

Input Queue Manager (IQM) -

The system routine that determines when a batch job is given to the CPU scheduler. IQM processes the RESOURCE control statement.

Invisible Package -

A hardware convention that contains the address and control information for the corresponding job.

Job -

A batch deck that is to be executed under control of the batch processor. A job begins with a job card and ends with a card having the digits 6, 7, and 9 multipunched in column 1.

Labeled Tape -

A magnetic tape with labels conforming to American National Standard X3.27-1969, Magnetic Tape Labels for Information Interchange.

Large Page -

128 512-word blocks; 65 536 contiguous 64-bit words.

Library -

A file of modules in a format produced by OLE that can be used to satisfy external references during loading.

Link Station -

A CYBER computer system controlled by the NOS/BE or SCOPE 3 operating system that is used to enter jobs into the CYBER 200 system and to control peripheral devices.

Local File -

A private file that is destroyed by the system after termination of the batch job or terminal session that created the file.

Map -

Part of the minus page of a virtual file that relates virtual addresses with physical mass storage addresses.

Mass Storage -

- (1) In a general sense, mass storage indicates disk-resident.
- (2) Specifically, a file management category that indicates no special processing after task termination.

Minus Page -

The first page of a virtual file used by the system to hold items such as the invisible package, input/output connector information, and maps of defined virtual space. Drop files contain a second minus page.

Nonprivileged -

A status that allows access to files owned by the same user number under which the task is running, to public files, and to authorized pool files.

Object Code File -

A file generated by compilation or assembly of a source language program that can be used by the loader to produce an executable file.

Output File -

A file destined for print or punch equipment.

Also, a generic term for a file being written, as opposed to an input file being read.

Ownership -

A category for each file that determines what nonprivileged tasks can access a mass storage file. Ownership categories are private, pool, and public. Private includes local and permanent files.

Pack File Index (PFI) -

A table of 16-word file index table entries that exists on each pack to control the files located on each of those packs.

Page -

The unit by which central memory is allocated; a block of contiguous 64-bit words.

Page Fault -

Reference by virtual address to a page not currently in central memory, causing a program interrupt and paging in.

Page Zero -

The second page of a virtual code file into which the CPU stores the task's register file when the task is not in the CPU.

Permanent File -

A private file that remains in the system after termination of the batch job or interactive session that creates it.

Physical Memory Address -

Address of a page in main memory. Also called physical address.

Pool -

A set of files created and maintained by a pool boss. More than one user number can access a pool.

Pool File -

An ownership category that indicates a file can be accessed by any privileged task and by any task running under a user number authorized by the pool boss.

Private File -

An ownership category that indicates a file can be accessed only by a task running under the user number under which the file is stored.

Privileged -

A status that allows access to all files in the system (except local files belonging to other users) and to most operating system functions.

Public File -

An ownership category that indicates a file can be accessed by all users.

Record -

The smallest logical set of data defined within an SIL file format.

Scalar -

A data item representing a single value that is processed by a scalar machine instruction (refer to Vector).

Scratch File -

A file that is destroyed upon termination of the task that creates it.

Security Level -

A level 0 through 7 established when a file is created. A user number is also associated with a security level. A user cannot access a file with a higher security level.

Segment -

An area of contiguous disk space allocated to a file.

Small Page -

512 contiguous 64-bit words.

Source File -

(1) A generic term for a file containing information used by a utility or other task whose specific meaning depends on the context of its use: the controllee file associated with a drop file, for instance, is termed the source file.

(2) In an Update utility context, a file produced by Update that would allow re-creation of a new program library on a subsequent creation run.

Station -

A peripheral processor linked to the CPU via an I/O channel. Each station performs a peripheral function for the CYBER 200 system such as providing access to on-line mass storage or to a front-end processor.

Suffix -

A letter A, B, C, or D that is associated with the user number under which a task executes. All batch jobs execute under suffix D; interactive tasks execute under the suffix specified by LOGON.

System Billing Unit (SBU) -

An installation-defined unit used for charging system resources. The unit might incorporate tape use/access, number of tape functions, number of disk accesses, number of pages transferred to or from disk, and CPU usage in microseconds. An example of SBU is time in microseconds of CPU use.

System Message -

The means by which the operating system and user tasks communicate with each other. System messages, which are formatted in Alpha words and Beta words, are described in volume 2 of the operating system reference manual.

System Time Unit -

An installation-defined unit used for allocating system resources. The unit might incorporate tape use/access, number of tape functions, number of disk accesses, number of pages transferred to or from disk, and CPU usage in microseconds. An example of STU is time in microseconds of CPU use.

Task -

An executable program.

Threshold value -

The maximum error code that a task can return without causing the batch processor to initiate job termination. The user sets the threshold value with the TV control statement.

User Number -

Six digits that identify a file owner or user of system resources. Only one task can be in execution for a given user number and suffix combination at one time.

USER1 -

A generic name for system routines that process card decks, print files, and punch files.

Vector -

A set of data items specified as a single operand for a vector machine instruction. Execution of the vector instruction processes each data item in the set.

Virtual Address -

Address that refers to virtual memory and is translated, through the page table, into a physical address.

Virtual Code File -

An executable file having a minus page as its first page and a page zero as its second page. The file must be created by the loader. A virtual code file is also called a controllee file. Contrast with Object Code File.

Virtual Memory -

A concept by which physical main memory can be addressed as if it were as large as needed.

Volume -

A reel of magnetic tape.

Word -

A division of central memory or mass storage corresponding to 64 bits. Bits are numbered 0 through 63 left to right.

Working Set -

The pages most frequently referenced during task execution. The size of the working set of a task determines when the task can be scheduled for CPU use.

80-Column Binary Card -

A punch card that is a representation of fifteen 64-bit words. No conversion occurs during card input or output. Also called an absolute binary card.

INDEX

- Abnormal job termination 3-9; 4-12
- Abnormal termination control 3-9
 - Error codes 3-10
 - Subroutine header 3-9
- Abort flag 3-4,9
- Absolute binary card format 3-2
- Access C-1
- Access modes 2-2
- Access station C-1
- Accessibility character 9-6
- Accessing an unattached permanent file
 - Control statement 4-3
 - Program call 9-6
- Account identifier C-1
- Accounting 3-7
- Active file 8-6
- ADDFILE Update directive 5-7
- Address format 1-3
- Advising the system of task virtual space needs 8-3
- Alternate control card file 4-25
- ANSI
 - Carriage control 2-7
 - Fixed length record format 2-6
 - Tape label formats 9-7
- ASCII
 - Carriage control 2-7
 - Character set 1-4; A-1
 - Coded cards 3-2
 - Debug directive 6-4
- Assigning tape files
 - Control statement 4-25
 - Program call 9-47
- ATC (refer to Abnormal termination control)
- ATTACH control statement 4-3
- Attaching
 - Permanent files
 - Control statement 4-3
 - Program call 9-6
 - Pools
 - Control statement 4-23
 - Program call 9-35
- AUDIT control statement 4-3
- BACK
 - Debug directive 6-3
 - Look directive 6-9
- Bank Update Table 8-24
- Banner cards 2-7
- BASE Look directive 6-9
- Batch
 - Deck 3-2
 - Job 1-4
 - Processing 3-3
 - Scheduling 3-8
 - Structure 3-3
 - Processor 3-1
 - Control statements 3-3; 4-1
 - System access 3-1
- BATCHPRO C-1
- BB request line 3-5
- BINARY file 3-4
- BIT Look directive 6-9
- BKPT Debug directive 6-5
- BKPTR Debug directive 6-5
- Blank common 4-17
- Blank compression and expansion 2-6
- Block 1-1; C-1
- Blocking type 2-5
- Bound explicit map 2-2
- Bound implicit map 2-2
- BP request line 3-5
- BREAK character 3-6
- Buffer memory 1-2
- BYE request line 3-6
- Bypass count 3-8
- Byte C-1
- CALL Update directive 5-8
- Calling conventions 8-2
- Calling parameters 8-3
- Card deck 3-2
- Card formats 3-2
- Card image file 5-1
- Carriage control 2-7
- Card reader ID card C-1
- Central memory 1-1
- Central operating system 1-2
- Central processor unit 1-1
- Changing
 - Accounting rate 8-48
 - File attributes
 - Control statement 4-31
 - Program call 9-9
 - File ownership
 - Control statement 4-14
 - Program call 9-25
 - FIT field values 9-50
 - Job memory limits 4-28
 - Object module linkages 4-17
 - Task large page limit 8-41
 - Terminal suffix 3-5,6
- Channel 1-2
- Character set 1-4
- Checking
 - I/O request status 9-10
 - Termination values 4-31
- Checkpoint 7-1
- CHKPNT call 7-1
- Closing files 9-10
- COMDECK Update directive 5-8
- COMMENT control statement 4-6
- Comments
 - In job dayfile 4-6
 - On control statement 4-1
- Common blocks 4-17,18
- COMPARE control statement 4-6
- Configuration 1-1
- Comparing file contents 4-6
- Compile file 5-4
- COMPILE Update directive 5-8
- Console 1-3
- Continuation character 4-1
- CONTINUE Debug directive 6-5
- Control statement 4-1
 - Continuation
 - Batch 4-1
 - Interactive 4-1
 - Format
 - Batch 4-1

- Interactive 4-1
 - Parameter format 4-1
 - Record 3-3
- Control word delimited record format 2-6
- Controllee 3-1
 - Chain 3-1
 - File 2-1
 - Termination status 8-19,46
- Controller (hardware) 1-2
- Controller (software) 3-1
- COPY control statement 4-7
- Copying
 - Bank Update Table 8-34
 - Invisible package 8-22
 - Minus page 8-27
 - Files 4-7
 - Pack label and file indices 8-27
 - Pool file indices 8-31
 - Private file indices 8-33
 - Public file indices 8-34
 - Register file 8-22
 - Statistics Buffer 8-38
 - Timecard Buffer 8-38
- Correction run 5-6
- Coupling station 1-2
- CPU (refer to Central processing unit)
 - Scheduler 3-8
- Creating
 - Controllee files 4-15
 - Library files 4-21
 - Local files 4-25
 - Modmerge files 4-21
 - Permanent files
 - Control statement 4-7
 - Program call 9-12
 - Pools
 - Control statement 4-23
 - Program call 9-35
 - Pool files
 - Control statement 4-14
 - Program call 9-25
 - Public files
 - Control statement 4-12,14
 - Program call 9-25
- Creation run 5-4
- CYBER 200 model descriptions 1-1
- Dayfile 3-4
- DDECIMAL Debug directive 6-3
- DEBUG control statement 6-1
- DEBUG directives 6-2
- Debugging routine versions 4-17
- Debugging utilities 6-1
- DEC Look directive 6-8
- DECIMAL Debug directive 6-4
- Decimal to hexadecimal conversion A-3
- DECK Update directive 5-9
- Decoding
 - Disk Status Table 8-5
 - Miscellaneous Table 8-5
 - Pack label 8-15
 - Permanent file indices 8-10
- DEFINE control statement 4-7
- DELETE Update directive 5-9
- Descriptor 8-3
- Destroying
 - Batch input file 8-16
 - Permanent files
 - Control statement 4-24
 - Program call 9-38
 - Pools
 - Control statement 4-23
 - Program call 9-36

- Pool files
 - Control statement 4-24
 - Program call 9-38
- Public files
 - Control statement 4-12
 - Program call 9-38
- Detaching
 - Permanent files
 - Control statement 4-28
 - Program call 9-45
 - Pools
 - Control statement 4-23
 - Program call 9-36
- Determining tape file status 9-53
- DFLOAT Debug directive 6-3
- Diagnostic messages B-1
- Directive C-1
- Directory file 4-9
- Disabling
 - Abnormal termination control 3-11; 8-16
 - Message interrupts 8-16
- Discarding FITs 9-44
- Disconnecting an initialized controllee 8-45
- Disk drive 1-2
- Disk files 2-1
- Disk Status Table 8-5
- DISPLAY
 - Debug directive 6-3
 - Look directive 6-8
- Disposition code 2-7
- Division 3-8
- Drawing time from repository 3-5
- DREG Debug directive 6-4
- Drop file 2-2
- DUMP control statement 6-10
- DUMPF control statement 4-9
- Dumping
 - Cumulative accounting file 8-18
 - Drop file information 6-10
 - Files 4-9
- Dynamic stack address 4-18
- EASCII Look directive 6-9
- EDEC Look directive 6-9
- EDITPUB control statement 4-12
- EFLOAT Look directive 6-9
- EHDEC Look directive 6-9
- EHEX Look directive 6-9
- EHFLOAT Look directive 6-9
- Enabling
 - Abnormal termination control 3-11; 8-19
 - Message interrupts 8-18
- END
 - Debug directive 6-5
 - Look directive 6-8
- End of file C-1
- End of information C-2
- End of partition codes 9-25
- EOF1 label 9-6
- EOV1 label 9-6
- EREG Debug directive 6-4
- Error messages B-1
- Error processing
 - SIL 8-2
 - System 3-9
- Error recovery limit 3-11
- Error severity
 - SIL 8-2
 - System 3-8
- Evicting local files 4-28
- Examining a mass storage file 6-6
- Excess bit count 9-11
- Executable format 4-18

- EXECUTE Debug directive 6-5
- Execution time file reassignment 8-42
- EXIT control statement 4-12
- Exit processing 3-9
- Expiration date 9-7
- Explicit I/O 2-1; 9-42,53; C-2
- Extension size 2-5

- F record format 2-6
- Family files 2-7; 3-5; C-2
- Fatal errors 3-9
- Free space 2-1
- File 2-1; C-2
 - Access controls 2-2
 - Access modes 2-2
 - Blocking 2-5
 - Duration 2-1
 - Extensions 2-5
 - Families 2-7; 3-4
 - Formats 2-5
 - I/O 2-1
 - Logical unit number 9-1
 - Names 2-1,3
 - Organization 2-5
 - Ownership 2-3
 - Passwords 2-2
 - Routing
 - Control statement 4-28
 - Program call 9-46
 - Search hierarchy 3-5
 - Security levels 2-2
 - Segments 2-5
 - Space allocation 2-5
 - Structure 2-5
- File Index Table C-2
- File Information Table (refer to FIT)
- FILEI 2-1; C-2
- FILES control statement 4-13
- FIT 9-1
 - Format 9-2
 - Limit 9-10
- Fixed-length blocking 2-5
- FLOAT
 - Debug directive 6-4
 - Look directive 6-8
- FORTTRAN compiler execution 3-4
- Front-end processor 1-1,2

- G request line 3-5
- Generating a FIT 9-15
- Getting
 - Accounting information
 - At terminal 3-5
 - Within program 8-19
 - Current date and time
 - At terminal 3-5
 - Within program 8-45
 - FIT field values 9-18
 - Large page limits 8-23
 - Message from
 - Controllee 8-23
 - Controller 8-25
 - Operator 8-25
 - Program state 3-5
 - Task
 - Information 8-28
 - Time limit 8-28
 - User number 8-29
 - Time task has used
 - At terminal 3-5
 - Within program 8-5
- GIVE control statement 4-14

- Glossary C-1
- GO file 3-4
- Granting access to a pool 9-36
- Grouping object modules 4-18
- Groups 2-6; C-2

- Hardware description 1-1
- HCD station (refer to High capacity disk station)
- HDEC Look directive 6-8
- HDR1 label 9-6
- HEX Look directive 6-8
- Hexadecimal number system 1-4
- Hexadecimal to octal conversion A-2
- Hexadecimal to decimal conversion A-3
- HFLOAT Look directive 6-8
- High capacity disk station 1-1

- I request line 3-6
- IDEC Look directive 6-8
- IDENT Update directive 5-9
- IDISPLAY Debug directive 6-3
- IDREG Debug directive 6-4
- IFLOAT Look directive 6-8
- IHDEC Look directive 6-8
- IHEX Look directive 6-8
- IHFLOAT Look directive 6-8
- Implicit I/O 1-1; 2-1; 9-29,30
- Initializing
 - Controllee 8-29
 - Controllee chain 8-30
- Input/output channels 1-1
- Input/output connector C-2
- INPUT file 3-3
- Input queue 3-8
- Input queue manager 3-3,8; C-2
- INSERT Update directive 5-10
- Interactive
 - Control statement execution 3-7
 - Job category 3-8
 - Processor 3-1
 - System access 3-4
 - Task call format 3-6
- Interrupt mode 3-11
- INTRACTV job category 3-8
- Invisible package 8-22
- IOC (refer to Input/Output Connector)
- IQM (refer to Input queue manager)

- JDEFAULT job category 3-8
- Job 1-4
 - Category 3-8; 4-27
 - Controller 3-1
 - Dayfile 3-4
 - Mode 1-2
 - Priority 3-8
 - Processing 3-3
 - Scheduling 3-8
 - Selection number 3-8
 - Structure 3-3
 - Termination 3-9

- KERNEL 1-2
- Keypunch conversion mode 3-2
- Keyword 8-2

- Labeled common 4-17
- Large page 1-1
- Large page limit
 - Job 4-27,28

- Task 3-6; 8-23,41
- Library file 4-15,17
- Linking object modules 4-17
- List file 5-4
- Listing
 - Attached pools 3-5
 - Controllee chain 8-34
 - Local file status 4-13
 - Permanent file information 4-3
 - Pools
 - Control statement 4-24
 - Program call 9-25
 - Time available 3-5
 - Time consumed 3-5
 - Users with pool access
 - Control statement 4-24
 - Program call 9-40
- Load map 4-18
- LOAD utility 4-15
 - Interactive use 3-7
- LOADPF control statement 4-18
- Local files 2-1
- Logging off from CYBER 200 OS 3-6
- Logging onto CYBER 200 OS 3-5
- Logical record formats 2-5
- LOGON command 3-5
- LOOK utility 6-6
- Magnetic tape files (refer to Tape files)
- Magnetic tape station 1-2
- Maintenance control unit 1-2
- Mapping in virtual space 9-29
- Mapping out virtual space 9-30
- Mass storage allocation 2-5
- Mass storage files 2-1
- Master control character 5-2
- MCU (refer to Maintenance control unit)
- Message direction control 8-39
- Message interrupts 3-7
- Microdrum 1-2,3
- Minus page 2-1; 4-6
 - Initialization 4-17
- Miscellaneous Table 8-5
- Modmerge file 4-15,21
- Monitor mode 1-2
- Multifile set 9-6
- Multifile tape volume 2-8
- Multivolume tape file 2-8
- Mutually exclusive parameters 8-3
- New program library 5-4
- No-operation keywords 8-3
- NORERUN control statement 4-21
- NUCLEUS 1-3
- Numbered common 4-17

- Object module 4-15
 - Library editing 4-21
- Octal to hexadecimal conversion A-2
- Old program library 5-4
- OLE control statement 4-21
- OP request line 3-6
- Opening files 9-31
- Operating system 1-2
- Operator message
 - From terminal 3-6
 - From program 8-43
- OPERATOR task 1-3
- Operator's console 1-3
- Option 8-2
- Optional parameters 8-3

- Overlays 1-4
- Output file 2-7
- OUTPUT file 3-4
- O26 punch code 3-2
- O29 punch code 3-2
- P request line 3-5
- PACCESS control statement 4-23
- Pack file index C-2
- Page fault C-3
- Page table 1-3
- Page zero 4-6; C-3
- PAGER 1-2
- Paging 1-1
- Paired parameter format 8-2
- PAKmmnnn file 4-10
- Partition delimiter 9-13
- PATTACH control statement 4-23
- PATTERN Look directive 6-9
- PCREATE control statement 4-23
- PDELETE control statement 4-23
- PDESTROY control statement 4-23
- PDETACH control statement 4-23
- Peripheral operating system 1-3
- Peripheral stations 1-1
- Permanent files 2-1
- PFI (refer to Pack file index)
- PFILES control statement 4-24
- Physical files 2-1
- Physical space 1-1
- Piece 2-6
- Pool
 - Boss 2-3
 - File utilities 4-21
 - Files 2-3
 - List 2-3
 - Entry format 9-37
 - Program calls 9-35
- Positioning a file 9-45,50
- PR request line 3-5
- Presetting labeled common 4-17
- Print control characters 2-8
- Print files 2-7
- PRINT Look directive 6-8
- Priority
 - Job 4-27
 - Task 3-6
- Private files 2-3
- Program states 3-6
- Pseudo file 4-10,19
- Public files 2-3
- Punch files 2-7
- PURDECK Update directive 5-10
- PURGE control statement 4-24
- PURGE Update directive 5-10
- Purging files 4-24

- Q5ADVISE 8-3
- Q5ATTACH 9-6
- Q5CHANGE 9-9
- Q5CHECK 9-10
- Q5CLOSE 9-10
- Q5CPUTIM 8-5
- Q5DAYFLE 3-4
- Q5DCDDST 8-5
- Q5DCDMSC 8-5
- Q5DCDPFI 8-10
- Q5DCDPLB 8-15
- Q5DEFINE 9-12
- Q5DESBIF 8-16
- Q5DISAMI 8-16
- Q5DISATI 8-16

Q5DMPACT 8-18
 Q5ENAMI 8-18
 Q5ENATI 8-19
 Q5ENDPAR 9-15
 Q5GENFIT 9-15
 Q5GETACT 8-19
 Q5GETCTS 8-19
 Q5GETFIL 9-18
 Q5GETFIT 9-22
 Q5GETHP 8-22
 Q5GETIRF 8-22
 Q5GETLP 8-23
 Q5GETMCE 8-23
 Q5GETMCR 8-25
 Q5GETMOP 8-25
 Q5GETMPG 8-27
 Q5GETN 9-25
 Q5GETP 9-25
 Q5GETPFI 8-27
 Q5GETTL 8-28
 Q5GETTN 8-28
 Q5GETUID 8-29
 Q5GIVE 9-25
 Q5INIT 8-29
 Q5INITCH 8-30
 Q5JOBFL 3-4
 Q5LFIPOL 8-31
 Q5LFIPRI 8-33
 Q5LFIPUB 8-34
 Q5LSTBUT 8-34
 Q5LSTCH 8-34
 Q5LSTSTB 8-38
 Q5LSTTCB 8-38
 Q5MAPIN 9-29
 Q5MAPOUT 9-30
 Q5MSGCTR 8-39
 Q5OPEN 9-31
 Q5PATACH 9-35
 Q5PCREAT 9-35
 Q5PDESTR 9-36
 Q5PDTACH 9-36
 Q5PGRACC 9-36
 Q5POOLS 9-36
 Q5PREACC 9-38
 Q5PURGE 9-38
 Q5PUSERL 9-40
 Q5PUTN 9-40
 Q5PUTP 9-41
 Q5READ 9-42
 Q5RECALL 8-39
 Q5REDUCE 9-44
 Q5RETFIT 9-44
 Q5RETURN 9-45
 Q5REWIND 9-45
 Q5RFI 8-40
 Q5ROUTE 9-46
 Q5RQUEST 9-47
 Q5RUNBIF 8-41
 Q5SETFIT 9-50
 Q5SETLP 8-41
 Q5SKIP 9-50
 Q5SNDMCE 8-42
 Q5SNDMCR 8-42
 Q5SNDMDF 8-43
 Q5SNDMJC 8-43
 Q5SNDMOP 8-43
 Q5SNDSTR 8-45
 Q5STATUS 9-53
 Q5TERM 8-45
 Q5TERMCE 8-45
 Q5TIME 8-45
 Q5VRACC 8-48
 Q5WRITE 9-53

R record format 2-6
 READ Update directive 5-10
 READCC control statement 4-25
 Reading
 Blocks 9-42
 Control statements from an alternate file 4-25
 Logical partitions 9-25
 Partial logical partitions 9-25
 Read-only file 2-2
 Record formats 2-5
 Record mark character 2-5
 Record mark delimited record format 2-6
 Redirecting messages 8-39
 Reducing file length 9-44
 Register file 1-4
 Copying 8-22
 Releasing mass storage space 4-28; 9-44,45
 Reloading files 4-18
 Removing user access to pool 4-23; 9-38
 Repository 3-6,8
 REQUEST control statement 4-25
 Requesting a file 4-25; 9-47
 Requesting and opening a file 9-18
 Required parameters 8-3
 RERUN control statement 4-27
 Resident system 1-2
 RESOURCE control statement 4-27
 Resource limits 3-8
 Restarting a program 7-2
 Retrieving FIT field values 9-22
 Return code 4-31; B-1
 RETURN control statement 4-28
 Return parameters 8-3
 Returning
 Control from an interrupt subroutine 8-40
 Files 4-28; 9-45
 Time to a repository 3-5
 Rewinding files 9-45
 ROLL
 Debug directive 6-3
 Look directive 6-9
 ROUTE control statement 4-28
 Routing files
 Control statement 4-28
 Program call 9-46

 S request line 3-5
 Satisfying externals 4-17
 Save table 8-43
 SBU (refer to Station buffer unit or
 System Billing Unit)
 (sc) (refer to Special character)
 Scalar processor 1-1
 SCANNER 1-3
 Scheduling 3-8
 Scratch files 2-1
 Search hierarchy 3-4
 SEARCH Look directive 6-7
 Security levels 2-2
 Segment allocation 2-5
 Segmentation 1-4
 Sending a message to the
 Controllee 8-42
 Controller 8-42
 Job controller 8-43
 Job dayfile
 Control statement 4-6
 Program call 8-43
 Operator
 From terminal 3-6
 From program 8-43
 Separator cards 3-2
 SEQ Look directive 6-9
 Sequential file organization 2-5

- Service station 1-1,2
- SET control statement 4-28
- Set identifier 9-6
- Setting
 - Exit path 4-12
 - Job resource limits 4-27
 - Norerun status 4-21
 - Program breakpoints 6-1
 - Rerun status 4-27
 - Task resource limits 3-6
 - Termination value 8-31
 - Threshold value 4-31
- SIL
 - Call format 8-2
 - Error message format B-1
 - Error message routing 8-2
 - Error messages B-15
 - Error processing 8-2
 - I/O calls 9-1
 - Non-I/O calls 8-1
- SIO (refer to SIL)
- Small page 1-1
- SNAP Debug directive 6-6
- Software description 1-2
- Source decks 5-1
- Source file maintenance utility 5-1
- Special character 3-5
- SRL (refer to SIL)
- SRM format (refer to System Record Manager format)
- Stand-alone parameter format 8-3
- Starting a previously initialized controllee 8-45
- STAT Debug directive 6-5
- Station buffer unit 1-2
- Station control unit 1-1,2
- Statistics Buffer 8-38
- Status code categories 8-2
- STEP debug directive 6-5
- Stepping through a program 6-1
- STORE card 3-1
- STU (refer to System Time Unit)
- SU request line 3-5
- Subroutine calling conventions 8-2
- Suffix 3-5
 - Request line 3-6
- Suspending task execution 8-39
- SWITCH control statement 4-31
- SYSLIB file 3-4; 4-17
- System Billing Unit 4-27; C-3
- System description 1-1
- System-generated drop file 2-2
- System Interface Language (refer to SIL)
- System message 8-1; C-3
- System operator's console 1-3
- System Record Manager format 2-6
- System second 4-27
- System Time Unit C-3

T request line 3-5

Tape

- Files 2-8
- File sets 9-6
- Label processing 9-6

Task 1-4; 3-1

- Execute line 3-6

TEMNEWPL file 5-2

Terminal login 3-5

Terminal message interrupts 3-6,7

- During ATC processing 3-11

Terminating a task and its controllees 8-45

Termination value 3-4,9; 4-31; B-1

Threshold value 3-4,9,12,31; B-1; C-3

Timecard Buffer 8-38

Time limit

- Job 4-27

- Task 3-6,11

• Index-6

- TPAKmmnn file 4-10
- TV control statement 4-31

U record format 2-6

U request line 3-6

Update

- Call parameters 5-2
- Card identification 5-5
- Control statement format 5-12
- Directives 5-7
- Utility 5-1

Undefined record format 2-6

UNFORMAT card 3-2

Unit record station 1-2

- Batch input 3-1

Unlabeled tape 9-6

User-generated drop file 2-2

User number 000000 2-3

USER1 9-25; C-3

- Errors B-1

Utilities 4-1; 5-1; 6-1

Variable rate accounting 4-12

Variable Rate Table 4-12

Vector C-3

Vector processor 1-1

Virtual code file format 4-18

Virtual files 2-1

VIRTUAL Look directive 6-9

Virtual memory 1-1

- Addressing 1-3

Virtual system tasks 1-2

Volume C-4

VOL1 label 9-6

W record format 2-6

Warning errors 8-2

Word C-4

WORD Look directive 6-8

Word size 1-4

Working set 1-2; C-4

Working set size limit

- Job 4-27,28

- Task 3-6

Write access 2-2

Write-temporary file 2-2

Writing

- Blocks 9-53

- Logical partitions 9-40

- Partial logical partitions 9-40

- Partition delimiters 9-15

YANK Update directive 5-11

YANKDECK Update directive 5-12

/ Update directive 5-12

? request line

405 card reader 1-2

415 card punch 1-2

512 line printer 1-2

6/7/8/9 card 3-1,2

65209-1 coupling station 1-2

657 tape units 1-2

659 tape units 1-2

7/8/9 card 3-2

7639 controller 1-2

80-column binary cards 3-2

819 disk drive 1-2

COMMENT SHEET

MANUAL TITLE: CDC CYBER 200 Operating System 1 Reference Manual,
Volume 1

PUBLICATION NO.: 60457000

REVISION: C

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ STATE: _____ ZIP CODE: _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

CUT ALONG LINE

AA3419 REV 4/79 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS

PERMIT NO. 8241

MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division

ARH219

4201 North Lexington Avenue

Saint Paul, Minnesota 55112



CUT ALONG LINE

FOLD

FOLD

CONTROL STATEMENT SUMMARY (contd)

Page No.

OLE, INPUT=lfm-list, $\left\{ \begin{array}{l} \text{NEWLIB=liblfm} \\ \text{MODMERGE=modlfm} \end{array} \right\}$, OMIT=sfm, mod-list, LIST=opt, OUTPUT=lfm/len.	4-21
PACCESS, poolname, USER= $\left\{ \begin{array}{l} \text{u-list} \\ * \end{array} \right\}$.	4-23
PATTACH, poolname.	4-23
PCREATE, poolname.	4-23
PDELETE, poolname, USER= $\left\{ \begin{array}{l} \text{u-list} \\ * \end{array} \right\}$.	4-23
PDESTROY, poolname.	4-23
PDETACH, poolname.	4-23
PFILES, $\left\{ \begin{array}{l} \text{poolname} \\ \text{USER= } \left\{ \begin{array}{l} \text{u-list} \\ * \end{array} \right\} \end{array} \right\}$.	4-24
PURGE, lfm-list, CL=pool, ST=xxx.	4-24
READCC, lfm.	4-25
REQUEST, lfm/len, ACCESS=acs, MNR=mnrm, MXR=mxr, PC=pc, RMK=rmk, RT=rt, TYPE=typ, SECURITY=lvl, PACK=packid, NOEXTEND, NOSEGMENT.	4-25
REQUEST, lfm, ACCESS=acs, MNR=mnrm, MXR=mxr, PC=pc, RMK=rmk, RT=rt, DEVTYPE=dev, DENSITY=den, LB=lbl, OWNER=ownid, TPMODE=tpm, VSN=vsnm.	4-25
RERUN.	4-27
RESOURCE, TL=t, JCAT=j, PRIORITY=p, WS=w, LP=lp.	4-27
RETURN, $\left\{ \begin{array}{l} \text{lfm-list} \\ * \end{array} \right\}$, UNLOAD=x.	4-28
ROUTE, lfm, DC=dc, $\left\{ \begin{array}{l} \text{DEF} \\ \text{SAVE} \end{array} \right\}$, IC=ic, FID=ffff, EC=ec, CM=cm, ST=st, TID=yyyyyyy, OT=ot, REP=n, DI=iiiiiii.	4-28
SET, WS=w, LP=lp.	4-28
SWITCH, oldlfm, newlfm, TYPE=typ, ACCESS=acs, RETENTION=days, DROP=dlen, MNR=mnrm, MXR=mxr, RT=rt, PC=pc, RMK=rmk, SFO=fom.	4-31
TV, value+	4-31
UPDATE, $\left\{ \begin{array}{l} \text{C} \\ \text{C=file} \end{array} \right\}$, D, F, $\left\{ \begin{array}{l} \text{I} \\ \text{I=lfm} \end{array} \right\}$, L=opt, $\left\{ \begin{array}{l} \text{N} \\ \text{N=lfm/\#nnn} \end{array} \right\}$, $\left\{ \begin{array}{l} \text{O} \\ \text{O=lfm/\#nnn} \end{array} \right\}$, $\left\{ \begin{array}{l} \text{P} \\ \text{P=lfm} \end{array} \right\}$, Q, $\left\{ \begin{array}{l} \text{S} \\ \text{S=lfm/\#nnn} \end{array} \right\}$, $\left\{ \begin{array}{l} \text{T} \\ \text{T=lfm/\#nnn} \end{array} \right\}$, 8, *=c, /=c.	5-12

CORPORATE HEADQUARTERS P.O. BOX 0 MINNEAPOLIS, MINNESOTA 55440

